Annotating and Predicting Non-Restrictive Noun Phrase Modifications

Gabriel Stanovsky **Ido Dagan**
Computer Science Department, Bar-Ilan University
gabriel.satanovsky@gmail.com
dagan@cs.biu.ac.il

Abstract

The distinction between restrictive and non-restrictive modification in noun phrases is a well studied subject in linguistics. Automatically identifying non-restrictive modifiers can provide NLP applications with shorter, more salient arguments, which were found beneficial by several recent works. While previous work showed that restrictiveness can be annotated with high agreement, no large scale corpus was created, hindering the development of suitable classification algorithms. In this work we devise a novel crowdsourcing annotation methodology, and an accompanying large scale corpus. Then, we present a robust automated system which identifies non-restrictive modifiers, notably improving over prior methods.

1 Introduction

Linguistic literature provides a large body of research distinguishing between two types of modifiers within noun phrases: (1) *Restrictive modifiers*, which constitute an integral part of the entity referenced by the NP, e.g., the underlined modifier in "*She wore the necklace that her mother gave her*", versus (2) Non-restrictive modifiers, which provide an additional or parenthetical information on an already definite entity, e.g., "*The speaker thanked president Obama who just came into the room*" (Huddleston et al., 2002; Fabb, 1990; Umbach, 2006).

The distinction between the two types is semantic in nature and relies heavily on the context of the NP. Evidently, many syntactic constructions can appear in both restrictive and non-restrictive uses. While the previous examples were of relative clauses, Figure 1 demonstrates this distinction in various other syntactic constructions.

Identifying and removing non-restrictive modifiers yields shorter NP arguments, which proved beneficial in many NLP tasks. In the context of abstractive summarization (Ganesan et al., 2010) or sentence compression (Knight and Marcu, 2002), non-restrictive modifiers can be removed to shorten sentences, while restrictive modification should be preserved.

Further, recent work in information extraction showed that shorter arguments can be beneficial for downstream tasks. Angeli et. al. (2015) built an Open-IE system which focuses on shorter argument spans, and demonstrated its usefulness in a state-of-the-art Knowledge Base Population system. Stanovsky et al. (2015) compared the performance of several off-the-shelf analyzers in different semantic tasks. Most relevant to this work is the comparison between Open-IE and Semantic Role Labeling (Carreras and Màrquez, 2005). Specifically, they suggest that SRL's longer arguments introduce noise which hurts performance for downstream tasks.

Finally, in question answering, omitting non-restrictive modification can assist in providing more concise answers, or in matching between multiple answer occurrences.

Despite these benefits, there is currently no consistent large scale annotation of restrictiveness, which hinders the development of automatic tools for its classification. In prior art in this field, Dornescu et. al (2014) used trained annotators to mark restrictiveness in a large corpus. Although they reached good agreement levels in restrictiveness annotation, their corpus suffered from inconsistencies, since it conflated restrictiveness annotation with inconsistent modifier span annotation.

The contributions of this work are twofold. Primarily, we propose a novel crowdsroucing anno-

tation methodology which decouples the binary (restrictive / non-restrictive) distinction from the modifier span annotation (Section 3). Following this methodology, in Section 4 we present a large scale annotated corpus, which will allow further research into the automatic identification of non-restrictive modification.

Additionally, we developed a strong automatic classifier, which learns from our new corpus (Section 5).[1] This classifier uses new linguistically motivated features which are robust enough to perform well over automatically predicted parse trees. While there is still much room for improvement, especially in some of the harder, more context-dependent, cases (most notably, prepositional and adjectival modifiers), our system provides an applicable means for identifying non-restrictive modification in a realistic NLP setting.

2 Background

In this section we cover relevant literature from several domains. In Section 2.1 we discuss the established linguistic distinction between restrictive and non-restrictive modification. Following, in Section 2.2 we discuss previous NLP work on annotating and identifying this distinction. Finally, in Section 2.3 we briefly describe the recent QA-SRL annotation paradigm (He et al., 2015), which we utilize in Section 3 as part of our annotation scheme.

2.1 Non-Restrictive Modification

Throughout the paper we follow Huddleston et. al's (2002) well-known distinction between two types of NP modifiers: (1) *Restrictive modifiers*, for which the content of the modifier is an integral part of the meaning of the containing NP, and, in contrast, (2) *Non-restrictive modifiers*, that present a separate, parenthetical unit of information about the NP.

While some syntactic modifiers (such as determiners or genitives) are always restrictive, others are known to appear in both restrictive as well as non-restrictive uses, depending on semantics and context (Huddleston et al., 2002; Fabb, 1990; Umbach, 2006). Among these are relative clauses, adjectival, prepositional, non-finite, and verbal modifiers. See Figure 1 for examples of different syntactic constructions appearing in both restrictive as well as non-restrictive contexts.

[1] Both classifier and corpus will be made available upon publication.

(RC1) The necklace *that her mother gave her*$^{+}$ is in the safe.
(RC2) The governor disagreed with the U.S ambassador to China *who seemed nervous*$^{-}$.
(NF1) People *living near the site*$^{+}$ will have to be evacuated.
(NF2) sheriff Arthur Lester, *standing against the wall*$^{-}$, looked tired.
(PP1) The kid *from New York*$^{+}$ won the lottery.
(PP2) The assassination of Franz Ferdinand *from Austria*$^{-}$ started WWI.
(AD1) The *good*$^{+}$ boys won.
(AD2) The water level rose a *good*$^{-}$ 12 inches.

Figure 1: Restrictive (marked in red and a plus sign) and non-restrictive (marked in blue and a minus sign) examples in different syntactic constructions, see elaboration in Section 2. Examples index: RC - Relative clause, NF - Non-finite clauses (Huddleston et al. [p. 1265]), PP - Prepositional modifiers, AD - Adjectival modifiers (Huddleston et al. [p. 528]).

For example, for *relative clause*, Huddleston et. al. [p. 1058] identifies both restrictive as well as non-restrictive uses (for which they use the terms *integrated* and *supplementary*, respectively). In the sentence marked (RC1), the highlighted relative clause is restrictive, distinguishing the necklace being referred to from other necklaces, while in sentence (RC2), the relative clause does not pick an entity from a larger set, but instead presents separate information about an already specified definite entity.

2.2 Non-Restrictive Modification in NLP

Syntactic and semantic annotations generally avoid the distinction between restrictive and non-restrictive modification (referred here as "restrictiveness" annotation).

The syntactic annotation of the Penn TreeBank (Marcus et al., 1993) and its common conversion to dependency trees (e.g., (de Marneffe and Manning, 2008)) do not differentiate the cases discussed above, providing the same syntactic structure for the semantically different instances. See figure 2 for an example.

Furthermore, prominent semantic annotations, such as PropBank (Kingsbury and Palmer, 2003), AMR (Banarescu et al., 2013), CCG (Hocken-

maier and Steedman, 2007), or FrameNet (Baker et al., 1998), also avoid this distinction. For example, PropBank does not differentiate between such modifiers, treating both types of modification as an integral part of an argument NP.

Two recent works have focused on automatically identifying non-restrictive modifications. Honnibal et al. (2010) added simple automated restrictiveness annotations to NP-modifiers in the CCGbank (Hockenmaier and Steedman, 2007). Following a writing style and grammar rule, a modifier was judged as non-restrictive if and only if it was preceded by a comma.[2] This annotation was not intrinsically evaluated, as it was carried as part of an extrinsic evaluation of a statistical parser.

Having similar goals to ours, Dornescu et al. (2014) sets the prior art at annotating and predicting non-restrictive modification. In the annotation phase, each of their trained annotators was asked to (1) Mark spans of words in the sentence as forming an NP modifier, and (2) Mark each span they annotated in (1) as either restrictive or non-restrictive, and specify its type from a predefined list (e.g., relative clause, adjectival modifier, etc.).

Their inter-annotator agreement on the first task (modifier span) was low, reaching pairwise F1 score of only 54.9%, possibly due to problems in the annotation procedure, as acknowledged by the authors. The second part of the annotation achieved better agreement levels, reaching kappa of 0.78 (substantial agreement) for type annotation and 0.51 (moderate agreement) for restrictiveness annotation.[3]

Following the creation of the annotated dataset, they developed rule based and machine learning classifiers. All of their classifiers performed only at about 47% F1, at least partly due to the inconsistencies in span annotation discussed above.

To conclude this survey, although an effort was made by Dorenscu et. al (2014), there is currently no available consistent corpus annotated with non-restrictive modifiers.

2.3 QA-SRL

Traditional Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005) is typically perceived as answering **argument role questions**, such as *who, what, to whom, when*, or *where*, regarding a target predicate. For instance, PropBank's ARG0 for the predicate **say** answers the question "*who* **said** *something?*".

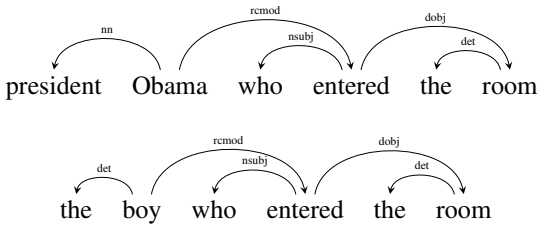

Figure 2: Restrictive (top) and non-restrictive (bottom) NP modifications receive the same representation in dependency trees. See Section 2.2.

QA-SRL (He et al., 2015) suggests that answering explicit role questions is an intuitive means to solicit predicate-argument structures from non-expert annotators. Annotators are presented with a sentence in which a target predicate[4] was marked, and are requested to annotate argument role questions, phrased using a restricted grammar, and corresponding answers.

For example, given the sentence "*President Obama who flew to Russia called the vice president*" and the target predicate **called**, an annotator can intuitively provide the following QA pairs: (1) *Who called?* **President Obama** and (2) *Whom did someone call?* **the vice president**.

In order to assess the validity of their annotation scheme, He et. al annotated a large sample of the PropBank corpus (1241 sentences) with QA-SRL, and showed high agreement with PropBank over this sample. In the following sections we make use of these explicit role questions for annotating non-restrictive modifiers.

3 Annotation Methodology

As mentioned in the Introduction, the first goal of this work is to assemble a large and consistent corpus, annotated with non-restrictive modifications. In this section, we present a crowdsourcing methodology which allows us to generate such corpus in a cost-effective manner (Section 3.2). As a preliminary step, we conducted a smaller scale expert annotation (Section 3.1), which will serve as a gold standard with which to test the crowsdsourced annotations.

[2]Notice that this is indeed the case in some of the non-restrictive examples in Figure 1.

[3]Note that the agreement for the first task is reported in F1 while the second task is reported in Cohen's kappa.

[4]Currently consisting of automatically annotated verbs.

3.1 Expert Annotation

Two researchers, with linguistics and NLP education, were presented with a sample of 219 modifiers of NPs in 100 sentences,[5] and were asked to annotate each modifier as either restrictive or non-restrictive, according to the linguistic definition presented in Section 2. Prior to annotating the expert dataset, the annotators discussed the process and resolved conflicts on a development set of 20 modifiers.

The annotators agreement was found to be high, reaching agreement on 93.5% of the instances, and κ of 84.2% . An analysis of the few disagreements found that the deviations between the annotators stem from semantic ambiguities, where two legitimate readings of the sentence led to disagreeing annotations. For example, in "*sympathetic fans have* **sent** *Ms. Shere copies of her recipes clipped from magazines over the years*", one annotator read the underlined modifier clause as restrictive, identifying particular recipes, while the second annotator read the modifier as non-restrictive, adding supplementary information on the sent recipes.

Finally, we compose the expert annotation dataset from the 207 modifiers agreed upon by both annotators. In the next section we use this dataset to evaluate the quality of our crowdsourced annotations.

3.2 Crowdsourcing Annotation Process

In our scheme, each annotation instance assigns a binary label (restrictive or non-restrictive) to a 4-tuple (s, v, p, m) – where m is a modifier of the noun phrase p, which is an argument of a verbal predicate v, in a sentence s. We incorporate v in our scheme in order to provide non-trained annotators with an argument role question (discussed in 2.3), as elaborated below.[6]

Consider, for example, the sentence s – "*the speaker* **thanked** *[President Obama who just entered the room]*". We want to annotate the restrictiveness value of the relative clause m (underlined), which modifies the matrix noun phrase p (bracketed), which is in turn an argument of a governing predicate v (in bold).

Our annotation procedure does not require the annotator to be familiar with the formal linguistic definition of restrictiveness. Instead, we use binary question-answering (true / false questions) as an intuitive formulation of non-restrictive modification. We present annotators with the argument role question pertaining to the argument NP, and ask whether this NP *without* the modifier gives the same answer to the argument role question as the original NP did.

In our example, an annotator is presented with the argument role question "*whom did someone thank?*" (which is answered by p), and is asked to decide whether the reduced NP, "*President Obama*", provides the same answer to the question as the full NP does. If the answer is positive (as in this case), we consider the modifier to be *non-restrictive*, otherwise we consider it to be *restrictive*.

As an example for the restrictive case, consider "*she* **wore** *[the necklace that her mother gave her]*", and the respective argument role-question "*what did someone wear?*". In this case, as opposed to the previous example, the reduced NP ("*the necklace*") does not refer to the same entity as the original NP, since we lose the specific identity of the necklace which was worn.

The intuition for this process arises from the linguistic definition for modifier restrictiveness. Namely, a restrictive modifier is defined as an integral part of the NP, and a non-restrictive modifier as providing supplementary or additional information about it. Therefore, in the restrictive case, omitting the modifier would necessarily change the meaning of the answer, while in the non-restrictive case, omitting it would not change the entity referenced by the full NP, and would therefore provide the same answer to the argument role question.

4 Corpus

In this section we describe the creation of a consistent human-annotated restrictiveness corpus, using the annotation process described in the previous section. We show this corpus to be of high quality by comparing it with the independent expert annotation. In Section 5 we use this corpus to train and test several automatic classifiers.

4.1 Data Collection

We use the dataset which He et. al (2015) annotated with Question-Answer pairs (discussed in Section 2.3), and keep their train / dev / test split

[5] These were taken at random from the development partition of the corpus described in Section 4.

[6] Our annotation currently covers the most common case of NPs which serve as arguments of verbal predicates.

Modifier Type	Identified By	#	Non-Restrictive	Agreement	
				κ	%
Adjectival	*pos* = JJ	684	41.36%	74.7	87.36
Prepositional	*pos* = IN / TMP / LOC	693	36.22%	61.65	85.1
Appositive	*rel* = APPO / PRN	342	73.68%	60.29	80
Non-Finite	*rel* = TO	279	68.82%	71.04	86.48
Verbal	*pos* = VB and not relative clause	150	69.33%	100	100
Relative clause	*pos* = VB and child *pos* = WP	43	79.07%	100	100
Total	**-**	**2191**	**51.12%**	**73.79**	**87**

Table 1: Corpus statistics by modifier types, which were identified by part of speech (*pos*) and dependency label (*rel*) (Section 4.1). The number of instances (#) and non-restrictiveness percentage refer to the full crowdsourced annotation. Agreement (Cohen's κ and percent of matching instances) is reported for the expert-annotated data (Section 4.2), between the expert and crowdsourced annotations.

into 744 / 249 / 248 sentences, respectively. This conveniently allows us to link between argument NPs and their corresponding argument role question needed for our annotation process, as described in previous section.

This dataset is composed of 1241 sentences from the CoNLL 2009 English dataset (Hajič et al., 2009), which consists of newswire text annotated by the Penn TreeBank (Marcus et al., 1993), PropBank (Kingsbury and Palmer, 2003), and NomBank (Meyers et al., 2004), and converted into dependency grammar by (Johansson and Nugues, 2008).

As mentioned in Section 3.2, each of our annotation instances is composed of a sentence s, a verbal predicate v, a noun phrase p, and a modifier m. We extract each such possible tuple from the set of sentences in the following automatic manner:

1. Identify a verb v in the gold dependency tree.
2. Follow its outgoing dependency arcs to a noun phrase argument p (a dependent of v with a nominal part of speech).
3. Find m, a modifying clause of p which might be non-restrictive, according to the rules described in Table 1, under the "Identified By" column. This filters out modifiers which are always restrictive, such as determiners or genitives, following (Huddleston et al., 2002), as discussed in Section 2. Notice that this automatic modifier detection decouples the span annotation from the restrictiveness annotation, which was a source for inconsistencies in Dornescu et al's annotation (Section 2.2).

This automatic process yields a dataset of 2191 modifiers of 1930 NPs in 1241 sentences. We note that our collection process ensures that the corpus correlates with the syntactic dependency annotation of the CoNLL 2009 shared task, and can therefore be seen as an augmentation of its modifier labels to include restrictiveness annotations.

In order to find the corresponding argument role question, we follow the process carried by He et. al; An argument NP is matched to an annotated Question-Answer pair if the NP head is within the annotated answer span. Following this matching process yields a match for 1840 of the NPs.

For the remaining 90 NPs we manually compose an argument role question by looking at the governing predicate and its argument NP. For example, given the sentence "*[The son of an immigrant stonemason of Slovenian descent] was* **raised** *in a small borough outside Ebensburg*", the predicate **raised** and the bracketed NP argument, we produce the argument role question "*Who was raised?*".

The corpus category distribution is depicted in Table 1, under column labeled "#". In later sections we report agreement and performance across these categories to produce finer grained analyses.

4.2 Crowdsourcing Annotation

We use Amazon Mechanical Turk[7] to annotate the 2191 modifiers for restrictiveness, according to the process defined in Section 3.2. Each modifier was given to 5 annotators, and the final tag was assigned by majority vote. We used the development set to refine the guidelines, task presentation, and

[7] https://www.mturk.com

the number of annotators.

Each annotator was paid 5c for the annotation of an NP, which in average provided 1.16 modifiers. This sets the average price for obtaining a single modifier annotation at $5 \cdot \frac{5}{1.16} = 21.5$c.

The agreement with the expert annotation and percentage of positive (non-restrictive) examples per category can be found in table 1, in the columns labeled "agreement". The labels are generally balanced, with 51.12% non-restrictive modifiers in the entire dataset (varying between 36.22% for prepositional modifiers and 79.07% for relative clauses).

Overall, the crowdsourced annotation reached good agreement levels with our expert annotation, achieving 73.79 κ score (substantial agreement). The lowest agreement levels were found on prepositional and appositive modifiers (61.65% and 60.29%).[8] Indeed, as discussed in Section 2, these are often subtle decisions which rely heavily on context. For example, the following instances were disagreed upon between our expert annotation and the crowdsourced annotation: In "*[Charles LaBella , the assistant U.S. attorney prosecuting the Marcos case], did n't* **return** *phone calls seeking comment*" (an appositive example), the experts annotated the underlined modifier as non-restrictive, while the crowdsource annotation marked it as restrictive. Inversely, in "*The amendment* **prompted** *[an ironic protest] from Mr. Thurmond*", the experts annotated the adjectival modifier as restrictive, while the crowdsource annotation tagged it as non-restrictive.

5 Predicting Non-Restrictive Modification

In this section we present an automatic system which: (1) Identifies NP modifiers in a dependency parser's output (as shown in Table 1, column "Identified By") and (2) Classifies each modifier as either restrictive or non-restrictive, based on the features listed in Table 2, and elaborated below.

5.1 Baselines

We begin by replicating the algorithms in the two prior works discussed in Section 2.2. This allows us to test their performance consistently against our new human annotated dataset.

Replicating (Honnibal et al., 2010) They annotated a modifier as restrictive if and only if it was preceded with a comma. We re-implement this baseline and classify all of the modifiers in the test set according to this simple property.

Replicating (Dornescu et al., 2014) Their best performing ML-based algorithm[9] uses the supervised CRFsuite classifier (Okazaki, 2007) over "standard features used in chunking, such as word form, lemma and part of speech tags". Replicating their baseline, we extract the list of features detailed in Table 2 (in the row labeled "chunking features").

5.2 Our Classifier

In addition to Dornescu et. al's generic chunking framework, we also extract features which were identified in the linguistic literature as indicative for non-restrictive modifications. These features are then used in the CRFsuite classifier (the same CRF classifier used by Donescu et al.) to make the binary decision. The following paragraphs elaborate on the motivation for each of the features.

Enclosing commas We extend Honnibal's et. al's classification method as a binary feature which marks whether the clause is both preceded and terminated with a comma. This follows from a well-known writing style and grammar rule which indicates that non-restrictive clausal modifiers should be enclosed with a comma.

Governing relative In the linguistic literature, it was posited that the word introducing a clausal modifier (termed *relative*) is an indication for the restrictiveness of the subordinate clause. For example, Huddleston. et al. (2002) [p. 1059] analyzes the word "that" as generally introducing a restrictive modifier, while a wh-pronoun is more likely to introduce non-restrictive modification. We therefore extract features of the word which governs the relative, such as the surface form, its lemma, POS tag, and more. The full list is shown under "Governing relative" in Table 2.

[8] While linguistic literature generally regards appositives as non-restrictive, some of the appositions marked in the dependency conversion are in fact misclassified coordinations, which explains why some of them were marked as restrictive.

[9] They also implement a rule-based method, named DAPR, which, when combined with the described ML approach surpasses their ML algorithm by $\sim$1.5% increase in F1. We could not find a publicly available implementation of this method.

Named entities As illustrated throughout the paper, modifiers of named entities tend to be non-restrictive. We run the Stanford Named Entity Recognizer (NER) (Finkel et al., 2005) and introduce a feature indicating the type of named entity (PERSON, ORG or LOC), where applicable.

Lexical word embeddings We include the pre-trained word embeddings of the modifier's head word, calculated by (Mikolov et al., 2013). These distributional features help the classifier associate between similar words (for example, if "good" is non-restrictive in some contexts, it is likely that "fine" is also non-restrictive within similar contexts).

Modifier type We add the automatically identified modifier type as a feature, to associate certain features as indicative for certain types of modifiers (e.g., enclosing commas might be good indicators for relative clause, while word embeddings can be specifically helpful for adjectival modifiers).

6 Evaluation

We evaluate each of the systems described in Section 5 on both gold trees as well as predicted trees, both sets provided in the CoNLL 2009 dataset (the predicted dependency relations were obtained using MaltParser (Nivre et al., 2007)). The gold setting allows us to test the performance of the systems without accumulating parser errors. In addition, it allows us to partition and analyze our dataset according to the gold modifier type. The predicted setting, on the other hand, allows us to evaluate our classifier in a real-world application scenario, given automatic parsing output.

6.1 Gold Trees

The results for each of the systems across our categories on the gold trees are shown in Table 3. Note that we regard non-restrictive modification as positive examples, and restrictive modification as negative examples. This is in line with the applicative goal of reducing argument span by removing non-restrictive modifiers, discussed in the Introduction. Switching the labels does not significantly change the numbers, since the corpus is relatively well balanced between the two labels (as can be seen in Table 1). Following are several observations based on an error analysis of these results.

Prepositional and adjectival modifiers are harder to predict All systems had more difficulties in classifying both of these categories. This reiterates the relatively lower agreement for these categories between the crowdsource and expert annotation, discussed in Section 4.2.

System	Feature Type	Description
Honnibal et. al	Preceding comma	w[-1] == ,
This paper	Chunking features (Dornescu et. al)	feats[head-1] feats[head] feats[head+1]
	Enclosing commas	true iff the clause is preceded and terminated with commas
	Governing relative	feats[parent-1] feats[parent] feats[parent+1]
	Prepositions	feats[pobj-1] feats[pobj] feats[pobj+1]
	NER	PERSON, ORGANIZATION, LOCATION
	Lexical word embeddings	Mikolov et al's 300-dimensional continuous word embeddings
	Modifier type	one of the types described in Table 1

Table 2: Features used for classification in each of the systems as described in Section 5. *head* - head of the modifier in the dependency tree. *parent* - parent of *head* in the dependency tree. *pobj* - object of the preposition, in case of prepositional *head*. *feats[i]* refers to extracting the following features from the word i: POS tag, lemma, is title, is all lower case, is all upper case, is beginning / end of sentence.

For clausal modifiers, preceding commas are good in precision but poor for recall As can be seen in Honnibal et. al's columns, a preceding comma is a good indication for a non-restrictive clausal modifier (all categories excluding adjectival or verbal modifiers), but classifying solely by its existence misses many of the non-restrictive instances.

(Dornescu et al., 2014) performs better on our dataset Their method achieves much better results on our dataset (compare 64% overall F1 on our dataset with their reported 45.29% F1 on their dataset). This speaks both for their method as a valid signal for restrictiveness annotation, as well as for the improved consistency of our dataset.

Modifier Type	#	Precision			Recall			F1		
		Honnibal	Dornescu	Our	Honnibal	Dornescu	Our	Honnibal	Dornescu	Our
Prepositional	135	.83	.67	.69	.1	.16	.41	.18	.26	.51
Adjectival	111	.33	.38	.59	.06	.06	.21	.11	.11	.31
Appositive	78	.77	.81	.82	.34	.93	.98	.47	.87	.89
Non-Finite	55	.77	.63	.64	.29	.97	.97	.42	.76	.77
Verbal	20	0	.75	.75	0	1	1	0	.86	.86
Relative clause	13	1	.85	.85	.27	1	1	.43	.92	.92
Total	412	.72	.72	**.73**	.19	.58	**.68**	.3	.64	**.72**

Table 3: Test set performance of the 3 different systems described in Sections 5 and 6 on gold trees from the CoNLL 2009 dataset, across the different categories defined in Section 4.

Features		**P**	**R**	**F1**
All		.73	.68	.72
Baseline	- comma	.72	.68	.7
	- chunking	.72	.66	.69
New	- governing relative	.74	.61	.67
	- prepositions	.73	.67	.7
	- word embeddings	.72	.69	.71
	- NER	.71	.68	.7
	- mod type	.74	.66	.7

Table 4: Feature ablation tests on gold trees. Each row specifies a different feature set – "All" specifies the entire feature set from Table 2, while each subsequent line removes one type of features.

System	**P**	**R**	**F1**
Candidate Extraction	.91	.93	.92
Honnibal	.71	.18	.29
Dornescu	.68	.53	.59
Our	**.69**	**.63**	**.66**

Table 5: Results on predicted trees. Candidate extraction measures the percent of correct modifiers identified in the predicted trees (shared across all of the classifiers). See Section 6.2.

Our system improves recall Overall, our system significantly outperforms both baselines by more than 8% gain in F1 score. Specifically, the numbers show clearly that we improve recall in the frequent categories of prepositional and adjectival modifiers. Furthermore, the results of an ablation test on our features (shown in Table 4) show that chunking and governing relative features provide the highest individual impact.

6.2 Predicted Trees

To test our classifier in a realistic setting we evaluate its performance on predicted dependency trees. To obtain the candidate modifiers, we use the same extractor presented in previous sections, applied on the predicted trees in the test section of the CoNLL 2009 dataset. We then apply the models trained on the gold train set of the same corpus.

For evaluation, we use the gold labels and compute (1) precision – the percent of predicted non-restrictive modifiers which match a gold non-restrictive modifier, and (2) recall – the percent of gold non-restrictive modifiers which match a predicted non-restrictive modifier. Note that this metric is strict, conflating both parser errors with our classifier's errors. The results are shown in Table 5. The first line in the table measures the performance of the automatic modifier extractor module versus the gold modifier extraction. This module is shared across all classifiers, as discussed in Section 5, and its performance imposes an upper bound on all the classifiers.

Both our and Dornescu's classifiers drop 5-6 points in F1, keeping the differences observed on the gold trees, while Honnibal et. al's simple comma-based classifier is less sensitive to parser errors, dropping only one point in F1.

This small drop stems from our classifiers largely relying only on the modifier head and its span for feature computation, generally ignoring parsing errors within the modifier subtree.

7 Conclusions and Future Work

We presented an end-to-end framework for restrictiveness annotation, including a novel QA-SRL based crowdsourcing methodology and a first consistent human-annotated corpus. Furthermore, we presented a linguistically motivated classifier, surpassing the previous baseline by 8% gain in F1.

Future work can use our annotated corpus to develop classifiers that deal better with prepositional and adjectival modifiers, which require deeper semantic analysis.

References

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL*, pages 86–90. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CONLL*, pages 152–164.

Marie-Catherine de Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Iustin Dornescu, Richard Evans, and Constantin Orasan. 2014. Relative clause extraction for syntactic simplification. *COLING 2014*, page 1.

Nigel Fabb. 1990. The difference between english restrictive and nonrestrictive relative clauses. *Journal of linguistics*, 26(01):57–77.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. In *Computational Linguistics*.

Matthew Honnibal, James R Curran, and Johan Bos. 2010. Rebanking ccgbank for improved np interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

Rodney Huddleston, Geoffrey K Pullum, et al. 2002. The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2003. Propbank: the next level of treebank. In *Proceedings of Treebanks and lexical Theories*, volume 3.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, pages 24–31.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open ie as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Carla Umbach. 2006. Non-restrictive modification and backgrounding. In *Proceedings of the Ninth Symposium on Logic and Language*, pages 152–159.