Journal of Artificial Intelligence Research 51 (2014) 133–164 Submitted 05/14; published 09/14

Text Rewriting Improves Semantic Role Labeling

Kristian Woodsend K.WOODSEND@ED.AC.UK
Mirella Lapata MLAP@INF.ED.AC.UK
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB

Abstract

Large-scale annotated corpora are a prerequisite to developing high-performance NLP systems. Such corpora are expensive to produce, limited in size, often demanding linguistic expertise. In this paper we use text rewriting as a means of increasing the amount of labeled data available for model training. Our method uses automatically extracted rewrite rules from comparable corpora and bitexts to generate multiple versions of sentences annotated with gold standard labels. We apply this idea to semantic role labeling and show that a model trained on rewritten data outperforms the state of the art on the CoNLL-2009 benchmark dataset.

1. Introduction

Recent years have witnessed increased interest in the automatic identification and labeling of the *semantic roles* conveyed by sentential constituents (Gildea & Jurafsky, 2002). The goal of the semantic role labeling task is to discover the relations that hold between a predicate and its arguments in a given input sentence (e.g., “who” did “what” to “whom”, “when”, “where”, and “how”).

(1) [Mrs. Yeargin]$_{\text{A0}}$ [gave]$_{\text{V}}$ [the questions and answers]$_{\text{A1}}$ [two days before the examination]$_{\text{TMP}}$ to [two low-ability geography classes]$_{\text{ARG2}}$.

In sentence (1), A0 represents the *Agent* or *giver*, A1 represents the *theme* or *thing given*, A2 represents the *Recipient*, TMP is a *temporal modifier* indicating when the action took place, and V determines the boundaries of the predicate. The semantic roles in the example are labeled in the style of PropBank (Palmer, Gildea, & Kingsbury, 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. Under the PropBank annotation framework each predicate is associated with a set of core roles (named A0, A1, A2, and so on) whose interpretations are specific to that predicate[1] and a set of adjunct roles such as *location* or *time* whose interpretation is common across predicates (e.g., *two days before the examination* in sentence (1) above).

This type of semantic information is shallow but relatively straightforward to infer automatically and useful for the development of broad coverage, domain-independent language understanding systems. Indeed, the analysis produced by existing semantic role labelers has been shown to benefit a wide spectrum of applications ranging from information extraction (Surdeanu, Harabagiu, Williams, & Aarseth, 2003) and question answering (Shen & Lapata,

1. More precisely, A0 and A1 have a common interpretation across predicates as *proto-agent* and *proto-patient* in the sense described by Dowty (1991).

Source	Target
1. The retreating guerrillas were soon pursued by the government forces.	Government forces soon pursued the retreating guerrillas.
2. A survey conducted by the Gallup Poll last summer indicated that one in four Americans takes cues from the stars or believes in ghosts.	A survey was conducted by the Gallup Poll last summer. It indicated that one in four Americans takes cues from the stars or believes in ghosts.
3. The examiner who was kind let the student finish his lunch.	The kind examiner let the student finish his lunch.
4. Because she didn't know the rules, she died.	She died, because she didn't know the rules.
5. Mexico City, the biggest city in the world, has many interesting archaeological sites.	Mexico City has many interesting archaeological sites.
6. The arrival of the train was unexpected.	The train's arrival was unexpected.

Table 1: Examples of syntactic rewriting.

2007), to machine translation (Wu & Fung, 2009) and summarization (Melli, Wang, Liu, Kashani, Shi, Gu, Sarkar, & Popowich, 2005).

Most SRL systems to date conceptualize the semantic role labeling task as a supervised learning problem and rely on role-annotated data for model training. Supervised methods deliver reasonably good performance, with F1-scores in the low eighties on standard test collections for English. They rely primarily on syntactic features (such as path features) in order to identify and classify roles. This has been a mixed blessing as the path from an argument to the predicate can be very informative but also quite complicated. Many paths through the parse tree are likely to occur a relatively small number of times (or not at all) resulting in very sparse information for the classifier to learn from. Even if the training data includes examples for a specific predicate and set of arguments, unless a test sentence contains them in the same syntactic structure, then as far as the classifier is concerned, the labeling of items within the two sentences is unrelated.

Our idea is to use use rewrite rules in order to create several syntactic variants for a sentence, thus alleviating the training requirements for semantic role labeling. Rewrite rules are typically synchronous grammar rules defining how a sequence of *source* terminals and nonterminals rewrites to a sequence of *target* terminals and nonterminals. Such rules are most often extracted from *monolingual* corpora containing parallel translations of the same source text (Barzilay & McKeown, 2001; Pang, Knight, & Marcu, 2003), *bilingual* corpora consisting of documents and their translations (Bannard & Callison-Burch, 2005a; Callison-Burch, 2007), or *comparable* corpora such as Wikipedia revision histories (Coster & Kauchak, 2011; Woodsend & Lapata, 2011). Examples of rewrites are given in Table 1. These include transforming passive to active sentences (see sentence pair (1) in Table 1), splitting a long and complicated sentence into several shorter ones (see (2) in Table 1), removing redundant parts of a sentence (see (3) in Table 1), reordering parts in a sentence (see (4) in Table 1), deleting appositives (see (5) in Table 1), transforming a prepositional phrase into a genitive (see (6) in Table 1), and so on.

We automatically extract syntactic rewrite rules from corpora and use them to generate multiple versions of role annotated sentences whilst preserving their original semantic roles. We therefore expand the training data with a wide range of syntactic variations for each predicate-argument combination and then learn a semantic role labeler on the expanded dataset. The approach we describe essentially increases the size of the training data by creating many different syntactic variations for different predicates and their roles.

Rewrite rules have been previously deployed in a variety of text-to-text generation applications ranging from summarisation (Galley & McKeown, 2007; Yamangil & Nelken, 2008; Cohn & Lapata, 2009; Ganitkevitch, Callison-Burch, Napoles, & Van Durme, 2011), to question answering (Wang, Smith, & Mitamura, 2007), information retrieval (Park, Croft, & Smith, 2011), simplification (Zhu, Bernhard, & Gurevych, 2010; Woodsend & Lapata, 2011; Feblowitz & Kauchak, 2013), and machine translation (Callison-Burch, 2008; Marton, Callison-Burch, & Resnik, 2009; Ganitkevitch, Cao, Weese, Post, & Callison-Burch, 2012). However, the application of text rewriting as a means of increasing the amount of labeled data available for model training is novel to our knowledge. We show experimentally, that syntactic transformations improve SRL performance beyond the state of the art when using the CoNLL-2009 benchmark dataset and the best scoring system (Björkelund, Hafdell, & Nugues, 2009). Importantly, our approach can be used in combination with any SRL learner or role-annotated data. Moreover, it is not specifically tied to the SRL task or the employed learning model and dataset. Rewrite rules could be used to expand the training data for other tasks that make use of syntactic features such as semantic parsing (Kwiatkowski, 2012) and textual entailment (Mehdad, Negri, & Federico, 2010; Wang & Manning, 2010).

In the following, we present an overview of related work (Section 2) and then describe how rewrite rules are automatically extracted and filtered for correctness (Section 3). Section 4 details our experiments and Section 5 reports our results.

2. Related Work

The idea of transforming sentences to make them more amenable to NLP technology dates back to Chandrasekar, Doran, and Srinivas (1996) who argue that simpler sentences would decrease the likelihood for an incorrect parse. To this end, they employ mostly hand-crafted syntactic rules aimed at splitting long and complicated sentences into simpler ones. Klebanov, Knight, and Marcu (2004) preprocess texts into Easy Access Sentences, i.e., sentences consisting of one finite verb and its dependents in order to facilitate information seeking applications such as summarization or information retrieval in accessing factual information. In a similar vein, Vickrey and Koller (2008) devise a large number of hand-written rules in order to simplify sentences for semantic role labeling. They present a log-linear model which jointly learns to select the best simplification (out of a possibly exponential space of candidates) and role labeling. Kundu and Roth (2011) use textual transformations for domain adaptation. Rather than training a new model on out-of-domain data, they propose to rewrite the out-of-domain text so that it is more similar to the training domain. They pilot their idea in semantic role labeling using hand-written rewrite rules and show that it compares favorably with approaches that retrain their model on the target domain.

Other work has focused on the idea of automatically expanding the data available for a given task without, however, applying any transformations. Fürstenau and Lapata (2012) combine labeled and unlabeled data by projecting semantic role annotations from a labeled source sentence onto an unlabeled target sentence. They find novel instances for classifier training based on their lexical and structural similarity to manually labeled seed instances. Zanzotto and Pennacchiotti (2010) increase the datasets for textual entailment by mining Wikipedia revision histories.

Contrary to previous work, we automatically extract general rewrite rules from various data sources including Wikipedia revision histories, comparable articles, and bilingual corpora. Given a sentence in the (semantic role) annotated data, we create several syntactic transformations, many of which may be erroneous. We only maintain for model training the transformations whose role labels are preserved under a syntactic rewrite. We identify which transformations are label-preserving automatically, without requiring any SRL-specific knowledge. Our approach differs from that of Vickrey and Koller (2008) in three important aspects: (a) we employ automatic rules which are not simplification specific, (b) we do not attempt to select the best rewrite, any transformations that preserve the gold standard role labels are used for training (c) we do not have a model that jointly rewrites sentences and labels their semantic roles; we only rewrite the training data which is then available to any model for the SRL task. Our work shares with that of Kundu and Roth (2011) the idea of transforming the sentences while preserving the gold standard role labels. However, we do not transform the test data to make it look like the training data. This unavoidably requires specialized knowledge of the differences between the two domains, which our more general model does not have.

As mentioned earlier, we use synchronous grammars to extract the set of possible syntactic rewrites. Synchronous context-free grammars (SCFGs; Aho & Ullman, 1969) are a generalization of the context-free grammar (CFG) formalism to simultaneously produce strings in two languages. They have been used extensively in syntax-based statistical MT (Wu, 1997; Yamada & Knight, 2001; Chiang, 2007; Graehl & Knight, 2004) and related generation tasks such as sentence compression (Galley & McKeown, 2007; Cohn & Lapata, 2009, 2013; Ganitkevitch et al., 2011), sentence simplification (Zhu et al., 2010; Feblowitz & Kauchak, 2013; Woodsend & Lapata, 2011), and summarization (Woodsend & Lapata, 2012). Rather than focusing on one type of transformation (e.g., simplification or compression), we learn the full spectrum of rewrite operations and then select rules appropriate for the task at hand. Furthermore, our results show that rewrite rules improve semantic role labeling performance across the board, irrespectively of the specific variant of synchronous grammar or corpus used. We experiment with conventional (weighted) SCFGs (Aho & Ullman, 1969) and tree substitution grammars (Eisner, 2003) and employ transformation rules extracted from Wikipedia revision histories (Zhu et al., 2010; Woodsend & Lapata, 2011) and bitexts (Ganitkevitch, Van Durme, & Callison-Burch, 2013).

3. Method

In this section we describe the general idea behind our algorithm and then move on to present our specific implementation. We define a *transformation* g to be a function that maps an example sentence s into a modified sentence s'. Let $\tilde{G}$ be the set of known transformation functions, $\tilde{G} = \{g_1, g_2, \ldots, g_{\tilde{n}}\}$. Suppose now that there are labels associated with example s. In the context of this paper, these are semantic role labels. Labels could be defined over spans of tokens, but here we use the CoNLL 2008–9 formalism where it is the head word of the span that is labelled. The transformation function is therefore a mapping between tokens t in sentence s to tokens t' in s'. We do not require that the mapping involves all the tokens of s or s', but we do require that the mappings are one-to-one.

A *label-preserving transformation* is a transformation g_i mapping from (some of the) tokens t in example s to tokens t' in s', such that the (correct) labels of t' are identical to the labels of its source tokens t for all the token mappings defined in g_i. In other words, those labels that could be preserved, have been preserved, and no others have been introduced. Let G be the set of label-preserving transformation functions, with $G \subset \tilde{G}$. The problem that we address in this paper is therefore two-fold: Firstly, to find automatically a set of possible transformation functions $\tilde{G}$ which due to its automated nature will unavoidably be an error-prone process. Secondly, to identify (again automatically) those transformations G which are actually label-preserving — more specifically, those transformations that rewrite a training instance s into s' through varying its syntactic structure, and yet preserve the semantic roles of those arguments that appear in the new version s'.

Algorithm 1 describes our approach which boils down to three steps: (a) extracting transformations, (b) refining transformations, and (c) generating and labeling an extended corpus. A standard gold annotated corpus is used to train an initial semantic role labeling model (see lines 1–2 in Algorithm 1). Meanwhile, a set of candidate transformation functions $\tilde{G}$ are extracted from some suitable comparable or parallel corpus (line 3). This full set of transformation functions is used to rewrite the gold corpus, creating a much extended corpus which inevitably will contain grammatically or semantically incorrect sentences. The extended corpus is next automatically labeled using the original SRL model after preprocessing through a normal SRL pipeline (whose details we discuss in Section 4.2), without knowledge of the transformation functions involved.

We could in theory use this extended corpus as the basis of training a further SRL model. However, it will contain many errors, and is unlikely to yield useful information to guide the model. One approach could be to manually correct the rewrites that have been generated automatically, but this would be very time and resource-intensive. Instead, we do the corrections automatically, and create an extended corpus where the rewrites do not impair the quality of the training data. We therefore learn which rules yield accurate rewrites, i.e., rewrites which preserve the labels of the gold-standard. Our intuition is that, given a large number of possible rewrites, the SRL model will in general label the accurate rewrites correctly and mis-label the erroneous sentences, due to it finding them more confusing. We thus compare the semantic role labels produced by the model with the labels for corresponding predicate-argument pairs in the gold corpus, and provide them as samples to train a binary classifier (here an SVM) which learns to predict which rewrites are likely to be successful and which are problematic (lines 11–19 in Algorithm 1).

Each rewritten sentence is classed as a positive sample if the SRL model is able to label the transformation to the same standard or better than it was able to label the original sentence, i.e. the labels that the SRL model predicts for the transformed sentence match those it predicted for the original, or have now been corrected with respect to the mapped gold labels. If, however, a semantic role is no longer predicted correctly, or missed, or an

Algorithm 1 Learn SRL model M_{extended} by extending a gold training corpus C_{gold} through transformation functions $\tilde{G}$.

$M_{\text{gold}} \leftarrow$ SRL model trained on C_{gold}
$C_{\text{model}} \leftarrow$ label C_{gold} using M_{gold}

Extract transformations:
$\tilde{G} \leftarrow$ all transformation functions that can be extracted from pairs of aligned sentences in comparable corpora

Refine transformations:
initialize SVM training data $D_{\text{SVM}} \leftarrow \emptyset$
for $s \in$ sentences in C_{gold} **do**
  for $g \in$ applicable transformations in $\tilde{G}$ **do**
    $s' \leftarrow$ rewrite s using g
    label s' using M_{gold}
    if SRL labels of s' match labels of s in C_{gold} or equivalent s in C_{model} **then**
      $y \leftarrow$ **true**
    else
      $y \leftarrow$ **false**
    end if
    add (s', y) to D_{SVM}
  end for
end for
train SVM using D_{SVM}
$G \leftarrow \{g \in \tilde{G} : g$ has positive SVM weight$\}$

Generate extended corpus:
initialize refined rewrite corpus $C_{\text{refined}} \leftarrow \emptyset$
for $s \in$ sentences in C_{gold} **do**
  for $g \in$ applicable transformations in G **do**
    $s' \leftarrow$ rewrite s using g
    project labels from s to s' using g
    add s' to C_{refined}
  end for
end for
$C_{\text{extended}} \leftarrow C_{\text{gold}} \cup C_{\text{refined}}$

Retrain SRL model:
$M_{\text{extended}} \leftarrow$ SRL model trained on C_{extended}
return M_{extended}

erroneous role introduced, this is classified as a negative sample, as such a sample is likely to harm the training of a new SRL model.

Once the SVM has identified the refined set of transformation functions G (line 20), these transformations are used to create an extended training corpus. This time, knowledge of the transformation function is involved to project the labels that correspond to the original gold corpus (lines 21–28). In the case of SRL, the labels describe the predicate and its arguments. This extended corpus supplements the original gold standard corpus (line 29), and the combination is then used to create a further SRL model (line 30).

It is worth noting that our method does not impinge on the actual process of learning an SRL model, as it is concerned with the preparation of training data. We therefore believe it can be applied to a range of SRL modeling approaches, and that gains in performance we achieve are largely orthogonal to those that could be made by improving other aspects of the learning process (see Section 5.3 for empirical evidence).

3.1 Learning Transformations

Conceptually a wide range of text-rewriting transformation functions could be included in the set $\tilde{G}$, such as paraphrasing, simplification or translation into another language. Here, we focus on transformation functions that can be expressed in synchronous context-free grammars (Aho & Ullman, 1969). Synchronous rules operate on parse tree constituents in a context-free manner, and typically modify the syntax. The transformations we consider can be sub-categorized into:

1. *Statement extraction.* Constituents of a sub-tree of the parse tree are identified, extracted from their context and rewritten as a complete sentence, typically shorter and simpler, although not necessarily so.

2. *Compression.* The original sentence is rewritten by compressing constituents of the parse tree, typically by deleting nodes.

3. *Insertion.* New elements are added to the parse tree. As significant chunks of new text would have semantic role information of their own, in practice these insertions are often additional punctuation to clarify the scope of phrases, or a simple structure such as "It is . . . ." to aid in statement extraction.

4. *Substitution.* Through using a lexicalized synchronous grammar, text can be replaced with new text, and paraphrases represented.

We obtain a set of possible transformations $\tilde{G}$ from monolingual comparable corpora drawn from Wikipedia and bitexts (see Section 4 for details). In the following we describe the grammar formalisms and resources we consider.

3.1.1 Transformations from Monolingual Corpora

We extract transformation rules from corpora that are only broadly comparable, using an unsupervised process. These corpora are constructed from Wikipedia revision histories, and comparable Wikipedia articles. As a result, it cannot be guaranteed that the aligned source and target sentences are truly related, nor can it be expected that the source sentence will fully generate the target sentence. In practice this means that in addition to not requiring a strictly synchronous structure over the source and target sentences, we cannot assume an alignment between source and target root nodes, or require a surjective alignment of all target nodes to nodes in the source parse tree. To be able to describe structural mismatches and non-isomorphic tree pairs (the grammar rules can comprise trees of arbitrary depth, and fragments can be mapped) we represent transformation functions using the synchronous tree substitution grammar formalism (Eisner, 2003).

A synchronous tree-substitution grammar (STSG) defines the space of valid pairs of source and target parse trees. Rules specify how to map tree fragments of the source parse tree into fragments in the target tree, recursively and free of context. Following Cohn and Lapata (2009), a STSG is a 7-tuple, $G = (\mathcal{N}_S, \mathcal{N}_T, \Omega_S, \Omega_T, P, R_S, R_T)$ where $\mathcal{N}$ are the non-terminals and Ω are the terminals, with the subscripts S and T indicating source and target respectively. P are the productions and $R_S \in \mathcal{N}_S$ and $R_T \in \mathcal{N}_T$ are the distinguished root symbols.

Typically, each production is a rewrite rule for two aligned non-terminals $X \in \mathcal{N}_S$ and $Y \in \mathcal{N}_T$ in the source and target:

$$\langle X, Y \rangle \to \langle \alpha, \gamma, \sim \rangle,$$

where α and γ are *elementary trees* rooted with the symbols X and Y respectively. While in a synchronous context free grammar α and γ would be limited to one level elementary trees, an STSG imposes no such limits and the elementary trees can be arbitrarily deep. A one-to-one alignment between the *frontier nodes* (non-terminal leaves of the elementary trees) in α and γ is specified by $\sim$.

In our experiments, we investigate two STSG variants, the strictly synchronous tree substitution grammar T3 (Cohn & Lapata, 2009), which was originally developed for the task of text compression, but does support a full range of transformation operations and the quasi-synchronous tree substitution grammar (QTSG) of Woodsend and Lapata (2011), which has been used in text simplification and summarization (Woodsend & Lapata, 2012). In T3 tokens are first aligned using a probabilistic aligner which has been initially provided with identity mappings for the entire vocabulary. In our experiments we used the Berkeley aligner (Liang, Taskar, & Klein, 2006), however any aligner with broadly similar output could have been used instead. Synchronous rules comprising trees of arbitrary depth are extracted from the pair of input CFG parse trees, consistent with the alignment. Across the complete corpus of aligned trees, T3 filters the extracted rules to provide the maximally general rule set, consisting of rules with the smallest depth, which are still capable of synchronously deriving any of the original aligned tree pairs. After removing identity rules, the resulting grammar forms the transformation functions $\tilde{G}$.

Unlike T3, QTSG works with only a partial alignment of tokens, based on identity. Non-terminal nodes in the parse trees are then aligned to be consistent with the token alignment. This can have the result that sections of both the source and target parse trees remain unaligned. Then, like T3, synchronous rules comprising trees of minimum necessary depth are extracted from the pair of input trees, consistent with the alignment, and as before, identity rules are removed to form $\tilde{G}$.

As an example, Figure 1 shows two comparable parse trees aligned at the token level. The synchronous rules extracted from this alignment by T3 and QTSG are shown in Table 2.

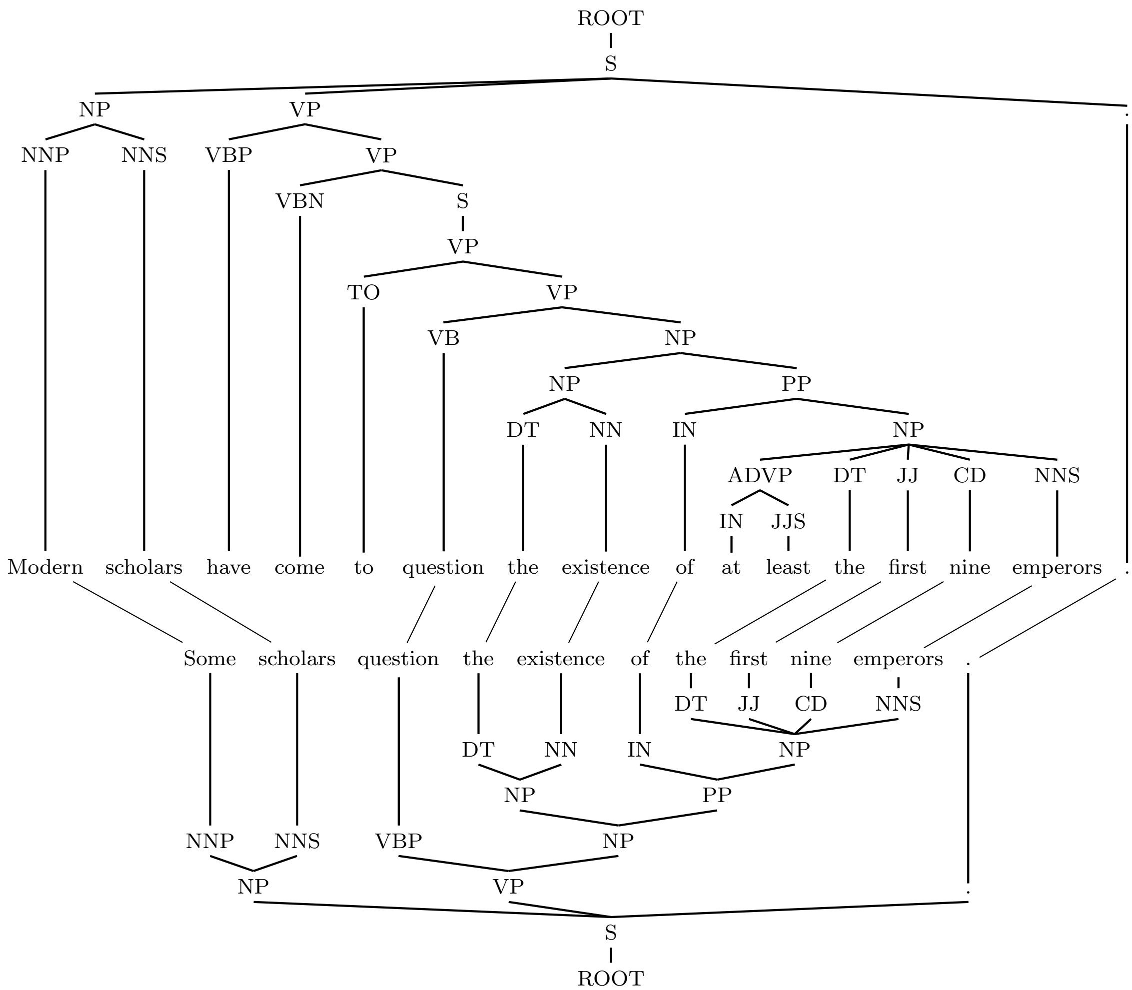

Figure 1: Example of sentence alignment showing source (above) and target (below) trees.

Where it is possible to extract rules from nodes at the child level, then the rules that T3 and QTSG extract at the parent level will be identical. In cases where a sub-tree has been compressed, (in the example, *have come to question* is compressed to *question*), QTSG extracts the full sub-tree until frontier nodes align, while T3 will extract several rules of the smallest depth.

3.1.2 Transformations from Bitexts

We also obtain transformation rules from the ParaPhrase DataBase (PPDB, Ganitkevitch et al., 2013), a collection of English (and Spanish) paraphrases derived from large bilingual parallel corpora. A variety of paraphrases (lexical, phrasal, and syntactic) are obtained following Bannard and Callison-Burch's (2005b) bilingual pivoting method.

Rules extracted by T3		
⟨S, S⟩	→	⟨[S NP$_{\boxed{1}}$ VP$_{\boxed{2}}$.$_{\boxed{3}}$], [S NP$_{\boxed{1}}$ VP$_{\boxed{2}}$.$_{\boxed{3}}$]⟩
⟨NP, NP⟩	→	⟨[NP [NNP Modern] NNS$_{\boxed{1}}$], [NP [DT Some] NNS$_{\boxed{1}}$]⟩
⟨VP, VP⟩	→	⟨[VP VBP$_{\boxed{\epsilon}}$ VP$_{\boxed{1}}$], [VP VP$_{\boxed{1}}$]⟩
⟨VP, VP⟩	→	⟨[VP VBN$_{\boxed{\epsilon}}$ [S VP$_{\boxed{1}}$]], [VP VP$_{\boxed{1}}$]⟩
⟨VP, VP⟩	→	⟨[VP TO$_{\boxed{\epsilon}}$ VP$_{\boxed{1}}$], [VP VP$_{\boxed{1}}$]⟩
⟨VP, VP⟩	→	⟨[VP [VB question] NP$_{\boxed{1}}$], [VP [VBP question] NP$_{\boxed{1}}$]⟩
⟨NP, NP⟩	→	⟨[NP NP$_{\boxed{1}}$ PP$_{\boxed{2}}$], [NP NP$_{\boxed{1}}$ PP$_{\boxed{2}}$]⟩
⟨NP, NP⟩	→	⟨[NP DT$_{\boxed{1}}$ NN$_{\boxed{2}}$], [NP DT$_{\boxed{1}}$ NN$_{\boxed{2}}$]⟩
⟨PP, PP⟩	→	⟨[PP IN$_{\boxed{1}}$ NP$_{\boxed{2}}$], [PP IN$_{\boxed{1}}$ NP$_{\boxed{2}}$]⟩
⟨NP, NP⟩	→	⟨[NP ADVP$_{\boxed{\epsilon}}$ DT$_{\boxed{1}}$ JJ$_{\boxed{2}}$ CD$_{\boxed{3}}$ NNS$_{\boxed{4}}$], [NP DT$_{\boxed{1}}$ JJ$_{\boxed{2}}$ CD$_{\boxed{3}}$ NNS$_{\boxed{4}}$]⟩
Rules extracted by QTSG		
⟨S, S⟩	→	⟨[S NP$_{\boxed{1}}$ VP$_{\boxed{2}}$.$_{\boxed{3}}$], [S NP$_{\boxed{1}}$ VP$_{\boxed{2}}$.$_{\boxed{3}}$]⟩
⟨NP, NP⟩	→	⟨[NP NNP$_{\boxed{\epsilon}}$ NNS$_{\boxed{1}}$], [NP [DT Some] NNS$_{\boxed{1}}$]⟩
⟨VP, VP⟩	→	⟨[VP VBP$_{\boxed{\epsilon}}$ [VP VBN$_{\boxed{\epsilon}}$ [S [VP TO$_{\boxed{\epsilon}}$ [VP VB$_{\boxed{1}}$ NP$_{\boxed{2}}$]]]]], [VP VBP$_{\boxed{1}}$ NP$_{\boxed{2}}$]⟩
⟨NP, NP⟩	→	⟨[NP NP$_{\boxed{1}}$ PP$_{\boxed{2}}$], [NP NP$_{\boxed{1}}$ PP$_{\boxed{2}}$]⟩
⟨NP, NP⟩	→	⟨[NP DT$_{\boxed{1}}$ NN$_{\boxed{2}}$], [NP DT$_{\boxed{1}}$ NN$_{\boxed{2}}$]⟩
⟨PP, PP⟩	→	⟨[PP IN$_{\boxed{1}}$ NP$_{\boxed{2}}$], [PP IN$_{\boxed{1}}$ NP$_{\boxed{2}}$]⟩
⟨NP, NP⟩	→	⟨[NP ADVP$_{\boxed{\epsilon}}$ DT$_{\boxed{1}}$ JJ$_{\boxed{2}}$ CD$_{\boxed{3}}$ NNS$_{\boxed{4}}$], [NP DT$_{\boxed{1}}$ JJ$_{\boxed{2}}$ CD$_{\boxed{3}}$ NNS$_{\boxed{4}}$]⟩

Table 2: Synchronous tree grammar rules extracted by T3 and QTSG from the aligned sentences in Figure 1. Boxed indices are short-hand notation for the alignment, $\sim$.

The intuition is that two English strings e_1 and e_2 that translate to the same foreign string f can be assumed to have the same meaning. The method then pivots over f to extract $\langle e_1, e_2 \rangle$ as a pair of paraphrases. An example is shown in Figure 2 (taken from Ganitkevitch et al., 2013). The method extracts a wide range of possible paraphrases some of which are unavoidably noisy due to inaccurate word alignments. Paraphrases are ranked by computing $p(e_1|e_2)$ as shown below:

$$p(e_2|e_1) \approx \sum_f p(e_2|f)p(f|e_1) \tag{2}$$

where $p(e_i|f)$ and $p(f|e_i)$ are translation probabilities estimated from the bitext (Koehn, Och, & Marcu, 2003).

Rewrite rules in PPDB are obtained using a generalization of the method sketched above to extract syntactic paraphrases (Ganitkevitch et al., 2011). Using techniques from syntactic machine translation (Koehn, 2010), SCFG rules are first extracted from English-foreign sentence pairs. For a foreign phrase the corresponding English phrase is found via the

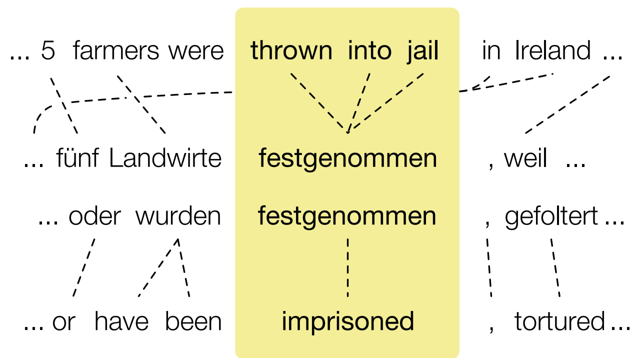

Figure 2: Paraphrases extracted via bilingual pivoting.

word alignments. This phrase pair is turned into a SCFG rule by assigning a left-hand side nonterminal symbol, corresponding to the syntactic constituent that dominates the English phrase. To introduce nonterminals into the right-hand sides of the rule, corresponding subphrases in the English and foreign phrases are replaced with nonterminal symbols. Doing this for all sentence pairs in a bilingual parallel corpus results in a translation grammar that serves as the basis for syntactic machine translation. A translation grammar can be converted into a paraphrase grammar as follows. Let r_1 and r_2 denote translation rules where the left-hand side nonterminals $\langle X, Y\rangle$ and foreign language strings γ match:

$$\begin{aligned} r_1 &= \langle X, Y\rangle \rightarrow \langle \alpha_1, \gamma, \sim\rangle \\ r_2 &= \langle X, Y\rangle \rightarrow \langle \alpha_2, \gamma, \sim\rangle \end{aligned} \tag{3}$$

A paraphrase rule r_p is then created by pivoting over f:

$$r_p = \langle X, Y\rangle \rightarrow \langle \alpha_1, \alpha_2, \sim\rangle \tag{4}$$

Although not shown in equations (3) and (4), the rules of the SCFG are associated with a set of features that are combined in a log-linear model to estimate the derivation probabilities.

3.1.3 Manual Transformations

Our experiments primarily make use of automatically learned transformations since these can be adapted to different tasks, domains or languages. However, for the proposed approach it is not necessary that transformation functions are acquired automatically — such functions could be also crafted by hand. We thus also investigated the effectiveness of rewrites generated by the system of Heilman and Smith (2010) (henceforth H&S), which uses a sophisticated hand-crafted rule-based algorithm to extract simplified declarative sentences in English from syntactically complex ones. These rules are similar to those engineered by Vickrey and Koller (2008) but deterministic in that they will only generate a unique rewrite for a given sentence. The algorithm operates on the standard phrase structure tree of an input sentence. It extracts new sentence trees from the input tree for the following: non-restrictive appositives and relative clauses; subordinate clauses with a subject and finite verb; and participial phrases that modify noun phrases, verb phrases, or clauses. In addition, the algorithm splits conjoined S, SBAR, or VP nodes, and extracts new sentence trees for each conjunct. Each output tree is further processed to move any leading prepositional phrases and quotations to be the last children of the main verb phrase, and the following are removed: noun modifiers offset by commas (non-restrictive appositives, non-restrictive relative clauses, parenthetical phrases, participial phrases), verb modifiers offset by commas (subordinate clauses, participial phrases, prepositional phrases), leading modifiers of the main clause (nodes that precede the subject).

Table 3 shows examples of rules extracted using the T3, QTSG and PPDB grammar formalisms applied to a sentence from the CoNLL dataset. The final column of Table 3 indicates whether the transformation could be classed as statement extraction, compression, insertion, or substitution. As reflected in the table, T3 captures compression transformations by deleting nodes in the parse tree; QTSG rules are a range of mainly syntactic transformations; and PPDB transformations are substitutions of words or short phrases.

3.2 Refining Transformations

As mentioned earlier, the transformation rules obtained from our synchronous grammars could be used to rewrite the gold standard sentences. Unfortunately, due to the nature of the corpora from which the rules are obtained and the automatic extraction process, many of the rules will contain errors which will impair rather than improve the quality of the training data. Our idea is to extrapolate which rules to trust by observing how the SRL labeler handles the rewritten sentences. If it has mis-labeled them, it is possible that the rewrite is not correct or that the original labels have not been preserved.

Each rewritten sentence is classed as a positive sample if the SRL model predicts the same labels for the transformed sentence as those it predicted for the original, or the labels have now been corrected with respect to the gold labels. If, however, a semantic role is no longer predicted correctly, or missed, or an erroneous role introduced, this is classified as a negative sample, as such a sample is likely to harm the training of a new SRL model. To capture the full impact of a candidate transformation function, a sentence is provided as a positive sample to the classifier only if all the labels (i.e., all predicates and arguments) from the source sentence have been successfully projected onto the rewrite. Table 4 shows examples of positive and negative samples of T3, QTSG, and PPDB rewrites. Note that no refining was used on the H&S outputs.

To decide which transformation function to include in the refined set, we used a linear kernel SVM (Vapnik, 1995) as a binary classifier, but other classifiers or indeed suitable statistical tests for contingency could be used. The input to the SVM learner is a set of l training samples $(x_1, y_1), \ldots, (x_l, y_l)$, $x_i \in R^n$, $y \in \{+1, 1\}$. x_i is an n dimensional feature vector representing the i^{th} sample, and y_i is the label for that sample. The learning process involves solving a convex optimization problem to find a large-margin separation hyperplane between positive and negative samples. In order to cope with inseparable data, some misclassification is allowed, the amount of which is determined by a parameter C, which can be thought of as a penalty for each misclassified training sample. From one

Grammar	Examples	Type
Original	Bell, based in Los Angeles, makes and distributes electronic, computer and building products.	
T3	Bell, based, makes and distributes electronic, computer and building products.	
	⟨PP, PP⟩ → ⟨[PP IN$_{\boxed{\epsilon}}$ NP$_{\boxed{\epsilon}}$], [PP]⟩	Comp
	Bell, based in Los Angeles, makes and distributes.	
	⟨NP, NP⟩ → ⟨[NP ADJP$_{\boxed{\epsilon}}$ NNS$_{\boxed{\epsilon}}$], [NP]⟩	Comp
	Based in Los Angeles, makes and distributes electronic, computer and building products.	
	⟨NP, NP⟩ → ⟨[NP NNP$_{\boxed{\epsilon}}$,], [NP]⟩	Comp
	Bell, based in Angeles's, makes and distributes electronic, computer and building products.	
	⟨NP, NP⟩ → ⟨[NP NNP$_{\boxed{\epsilon}}$ NNP$_{\boxed{1}}$], [NP NNP$_{\boxed{1}}$ [POS 's]]⟩	Comp
QTSG	Bell makes and distributes electronic, computer and building products.	
	⟨NP, NP⟩ → ⟨[NP NP$_{\boxed{1}}$, VP$_{\boxed{\epsilon}}$,], [NP NP$_{\boxed{1}}$]⟩	Comp
	It makes and distributes electronic, computer and building products.	
	⟨S, S⟩ → ⟨[S NP$_{\boxed{\epsilon}}$ VP$_{\boxed{1}}$.$_{\boxed{2}}$], [S [NP It] VP$_{\boxed{1}}$.$_{\boxed{2}}$]⟩	Ins
	Bell was based in Los Angeles.	
	⟨NP, S⟩ → ⟨[NP NP$_{\boxed{1}}$, VP$_{\boxed{2}}$,], [S NP$_{\boxed{1}}$ [VP [VBD was] VP$_{\boxed{2}}$] .]⟩	Ext
	Bell, based in Los, makes and distributes electronic, computer and building products.	
	⟨NP, NP⟩ → ⟨[NP NNP$_{\boxed{1}}$ NNP$_{\boxed{\epsilon}}$], [NP NNP$_{\boxed{1}}$]⟩	Comp
	Los Angeles makes and distributes electronic, computer and building products.	
	⟨NP, NP⟩ → ⟨[NP NP$_{\boxed{\epsilon}}$, [VP VBN$_{\boxed{\epsilon}}$ [PP IN$_{\boxed{\epsilon}}$ NP$_{\boxed{1}}$]] ,], [NP NP$_{\boxed{1}}$]⟩	Comp
PPDB	Bell, founded in Los Angeles, makes and distributes electronic, computer and building products.	
	⟨VP, VP⟩ → ⟨[VP [X based] PP$_{\boxed{1}}$], [VP [X founded] PP$_{\boxed{1}}$]⟩	Sub
	Bell, building in Los Angeles, makes and distributes electronic, computer and building products.	
	⟨VP, VP⟩ → ⟨[VP [X based] IN$_{\boxed{1}}$ NP$_{\boxed{2}}$], [VP [X building] IN$_{\boxed{1}}$ NP$_{\boxed{2}}$]⟩	Sub
	Bell, based during Los Angeles, makes and distributes electronic, computer and building products.	
	⟨VP, VP⟩ → ⟨[VP VBN$_{\boxed{1}}$ IN$_{\boxed{\epsilon}}$ NP$_{\boxed{2}}$], [VP VBN$_{\boxed{1}}$ [X during] NP$_{\boxed{2}}$]⟩	Sub

Table 3: Examples of transformation rules extracted using T3, QTSG and PPDB grammar formalisms, applied to the sentence marked Original. The final column indicates whether the rule is statement extraction (Ext), compression (Comp), insertion (Ins) or substitution (Sub). As before, boxed indices are short-hand notation for the alignment, $\sim$.

view (the *dual problem*), the result is a set of Support Vectors, the associated weights α_i, and a constant b. From another view (the *primal problem*), the result is a vector w that defines the separation hyperplane, with a dimension that depends on the particular kernel

Original	Bell, based in Los Angeles, makes and distributes electronic, computer and building products.	
T3	Bell, based, makes and distributes electronic, computer and building products.	+
	Bell, based in Los Angeles, makes and distributes.	+
	Based in Los Angeles, makes and distributes electronic, computer and building products.	+
	Bell, based in Angeles's, makes and distributes electronic, computer and building products.	–
QTSG	Bell makes and distributes electronic, computer and building products.	+
	It makes and distributes electronic, computer and building products.	+
	Bell was based in Los Angeles.	+
	Bell, based in Los, makes and distributes electronic, computer and building products.	–
	Bell, based in Angeles, makes and distributes electronic, computer and building products.	+
	Los Angeles makes and distributes electronic, computer and building products.	–
PPDB	Bell, founded in Los Angeles, makes and distributes electronic, computer and building products.	–
	Bell, building in Los Angeles, makes and distributes electronic, computer and building products.	–
	Bell, based during Los Angeles, makes and distributes electronic, computer and building products.	–
H&S	Bell makes. Bell distributes. Bell is based in Los Angeles.	

Original	For its employees to sign up for the options, a college also must approve the plan.	
T3	For it, a college also must approve the plan.	–
	A college also must approve the plan.	+
	For its employees to sign up for the options, this a also must approve this the.	–
	For its employees to sign up for, a college also must approve the plan.	+
QTSG	For its employees to sign up for the options, a college also must approve.	–
	For its employees to sign up for the options, a college also must approve plan.	+
	For its employees to sign up for all of the options, a college also must approve the plan.	–
PPDB	For its employees to sign up for the options, a college also must adopt the plan.	+
	For its employees to sign up for the options, a college also must agree to the plan.	–
	For its employees to sign up for the options, a college also must endorse the plan.	+
	For its employees to sign up for the options, a college also needs to approve the plan.	–
H&S	A college must approve the plan for its employees to sign up for the options.	

Original	That went over the permissible line for warm and fuzzy feelings.	
T3	That went over the permissible line for feelings.	+
	That went over for warm and fuzzy feelings.	–
	That went over it for it.	–
	That went.	+
QTSG	That went over the line for warm and fuzzy feelings.	+
	That went over the permissible line for feelings.	–
	That went over permissible for warm and fuzzy feelings.	–
PPDB	That went over the permissible line for hot and fuzzy feelings.	–
	That went during the permissible line for warm and fuzzy feelings.	+
H&S	That went over the permissible line for warm and fuzzy feelings.	

Table 4: Examples of rewrites generated by T3, QTSG, and PPDB for a source sentence (Original) from the CoNLL-2009 training set. Symbols +/– indicate whether the sample was classified as positive (i.e., argument label preserving) and forms part of extended training corpus, or not.

function used in the SVM. In the case of the linear kernel function, w is n–dimensional, as the feature vectors, and there is a straight-forward relationship $w = \sum_{j=1}^{l} y_j \alpha_j x_j$ between primal and dual variables, effectively assigning weights to the explicitly specified features. Other kernel functions allow for interaction between variables. For instance when using binary valued features, a degree–2 polynomial kernel function implies that the classifier considers all available pairs of features as well.

We used the identity of the transformation functions involved as the features of each sample, so the size of the feature space $n = \|\tilde{G}\|$, and features were binary-valued. Other features could be easily incorporated in this setting, perhaps capturing information on the structure of the source sentence or the transformation function, and this might achieve good results in conjunction with a polynomial kernel, but we did not pursue this avenue further. Instead we used a linear kernel, and due to the simple structure of our features, the SVM assigned a weight to each transformation function independent of the source sentence. We chose which transformation functions should form the refined set based on whether their corresponding weight was above a global threshold value, and we set the threshold value by maximizing the performance of the resulting SRL model on the development set.

3.3 Labeling the Extended Corpus

Once the SVM has identified the refined set of transformation functions G, these transformations are used to create an extended training corpus. Using the alignment information in the transformation functions to trace the position of tokens from the original sentence to the rewrite, the semantic role labels of the gold corpus are projected onto the corresponding predicate-argument pairs in the rewritten corpus. Assuming the SVM has correctly identified the transformation function involved as indeed label-preserving, and that the transformation functions can be applied in the current context, the semantic role labeling of the rewrite will now be of the same quality standard as the source. Both conditions are however unlikely to be true, resulting in a degradation in the quality of the rewrite corpus. The corpus of rewrites is appended to the original gold standard corpus to create a new larger training corpus, which is then used to create a further SRL model.

4. Experimental Setup

In this section we present our experimental setup for assessing the performance of our approach. We give details on the corpora and grammars we used to create the transformations, and model parameters used to identify those that preserve labels. We explain how an existing SRL system was modified through our approach, and how we evaluated the effects of increasing the training data with our transformations.

4.1 Grammar Extraction

We extracted synchronous grammars from two monolingual comparable corpora drawn from Wikipedia. A corpus of 137,362 aligned sentences created by pairing Simple English Wikipedia with English Wikipedia (Kauchak, 2013). And a corpus of 14,831 paired sentences from comparing consecutive revisions of articles in Simple English Wikipedia (Woodsend & Lapata, 2011). These corpora provide a large repository of monolingual, comparable

Grammar	Aligned	Revisions
T3	13,562	5,386
QTSG	3,875	669

Table 5: Non-identical rules extracted from each Wikipedia corpus, with rules appearing only one or two times removed.

sentences, taken from real-world writing. Advantageously, Simple English Wikipedia encourages contributors to employ simpler grammar than the ordinary English Wikipedia; the corpora therefore naturally contain many examples of syntactic variation such as reordering and sentence splitting, as well as paraphrasing and changes to content. Table 5 lists the number of non-identical rules each grammar formalism extracted from the Wikipedia corpora, once rules with an instance count of only one or two were removed.

In addition to these grammars extracted from Simple English Wikipedia, we worked with the monolingual synchronous grammar included in the Paraphrase Database (Ganitkevitch et al., 2013), where paraphrases were extracted from bilingual parallel corpora. The English portion of PPDB contains over 220 million paraphrase pairs, including 140 million paraphrase patterns capturing syntactic transformations with varying confidence. To form a synchronous grammar, we used highest scoring 585,000 paraphrases from the subset of constituent syntactic paraphrases (where all nonterminals were labeled with Penn Treebank constituents).

4.2 Semantic Role Labeler

The method presented in this paper crucially relies on a semantic role labeler for refining the transformations and performing the semantic analysis in general. We used the publicly available system of Björkelund et al. (2009). Out of the those that competed in the CoNLL-2009 SRL-only challenge, it was ranked first for English language, and second overall. To the best of our knowledge, this system represents the state-of-the-art for English SRL parsing. The system architecture consists of a four-stage pipeline of classifiers: for predicate identification (although this module is not required in evaluation), for predicate sense disambiguation, a binary classifier for argument identification, and finally argument classification using a multiclass classifier. Beam search is used to identify the arguments of each predicate and to label them, according to local classifiers using features which relate mainly to dependency parse information linking predicates to potential arguments and their siblings. In addition, a global reranker can be used to select the best combination of candidates (see Section 5 for details). The SRL system requires tokenized input with lemma, POS-tag and dependency parse information. This information was already provided in the gold-standard training corpus (see immediately below). To create equivalent information for the transformed text and evaluation files, we used the MATE-TOOLS pipeline (Björkelund et al., 2009), retrained (like the SRL model itself) on just the training partition of the data.

We used the English language benchmark datasets from the CoNLL-2009 shared task to train and evaluate the SRL models. We identified and labeled semantic arguments for nouns

Corpus	Sentences	Tokens
Training	39,272	958,174
+ H&S	55,474	909,358
+ PPDB	238,732	7,071,550
+ T3	203,941	4,701,688
+ QTSG	500,627	9,623,471
+ T3 + QTSG	704,561	14,325,166
+ PPDB + T3	442,666	11,773,245
+ PPDB + QTSG	739,352	16,695,028
+ PPDB + T3 + QTSG	943,286	21,396,723
Development	1,334	33,368
Test in-domain	2,399	57,676
Test out-of-domain	425	7,207

Table 6: Statistics on corpora used to train and evaluate the SRL models.

and verbs (Hajič, Ciaramita, Johansson, Kawahara, Martí, Màrquez, Meyers, Nivre, Padó, Štěpánek, Straňák, Surdeanu, Xue, & Zhang, 2009). We used the training, development, test and out-of-domain test partitions as they were provided, and some statistics on these data sets are shown in Table 6. Specifically, we show the increase on the training data effected by our method when using transformations obtained from T3, QTSG, PPDB, and their combinations. For comparison we also use the manual transformations available from Heilman and Smith (2010). To train the SRL model (and also the previous stages in the NLP pipeline), we used data from the training partition only, while the development partition was used to identify the best subset of G transformations[2].

We used LibLinear (Fan, Chang, Hsieh, Wang, & Lin, 2008) to train the SVM, and the hyper-parameters of the SVM were tuned by cross-validation on the training set to maximise the area under ROC curve, using the automatic grid-search utility of the python package scikit-learn (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, & Duchesnay, 2011). An assessment of the cross-validation accuracy (in terms of F1 score and area under ROC curve) of the SVM for each grammar is shown in Table 7. The results show that PPDB rewrites are the most accurate to employ, perhaps because the rules are the most heavily lexicalized of all the grammars. T3 grammar is the most unpredictable to use, although the SVM scores considerably higher than chance.

Test sets were used solely for evaluation, making use of the indicators in the data files as to which words were argument-bearing predicates. Results were generated using the CoNLL-2009 evaluation script unmodified. We only report results on semantic roles (i.e., not in combination with syntactic dependencies which tends to yield higher scores) using both the in-domain and out-of-domain evaluation data. In the evaluation script, semantic propositions are evaluated by converting them to semantic dependencies between

2. The result of this re-training was that the performance reported here is worse than for the models available on the mate website, which have been trained on all partitions of the CoNLL-2009 data (training, development and test).

Grammar	F1	Area under ROC
PPDB	0.85	0.82
T3	0.67	0.61
QTSG	0.78	0.72

Table 7: Statistics on the SVM's performance for each grammar, obtained through cross-validation on the training set.

a predicate and each of its arguments, and labeling the dependency with the labels of the corresponding argument. Additionally, a dependency is created from a virtual root node to each predicate and labeled with the predicate sense. To be comparable with other published results, in general we report the scores that combine predicate sense and argument role label predictions. In Tables 12, 13 and 14, however, we focus on arguments only, and remove the predicate sense scores.

5. Results

In this section we provide empirical evidence on the performance of our approach. Our experiments were primarily designed to answer the following questions. Does text rewriting generally improve SRL performance? Does it matter which transformation rules to use, i.e., are some rules better than others? Are the transformation rules useful on out-of-domain data? Which SRL labels are mostly affected by rewriting? Does performance vary depending on the size of the original training data? Are the results sensitive to the learner being employed? We first examine the effect of different (transformation) grammars on the SRL task both on in-domain and out-of-domain test data, and then move on to assess which labels are mostly affected by our method. Finally, we present results on the effect of combining our approach with a global reranker and training with different-sized datasets.

5.1 Transformation Rules Improve F1 Across the Board

Table 8 (left half) shows SRL performance (measured in terms of precision, recall, and F1) on the in-domain CoNLL-2009 test set. For the training corpora rewritten by the H&S system, the T3, QTSG, and PPDB grammars, all of the resulting SRL models significantly ($p < 0.01$) improve over a model trained on the original corpus. We used stratified shuffling (Noreen, 1989) to examine whether differences in F1 were significant (Padó, 2006). Recall shows the largest increase, particularly with the acquired synchronous grammars, indicating that the increased training data is resulting in better coverage. Generally this is not at the expense of precision which in all cases apart from PPDB has increased as well. Significant gains are also seen in the acquired grammars compared to the H&S system, with the exception of T3 where there is greater variation in its performance.

We also combined the rewrites produced by the different grammars (see T3+QTSG, PPDB+T3, PPDB+QTSG and PPDB+T3+QTSG in Table 8) but this did not significantly improve performance over the individual grammars (although still significantly better than the original model and the H&S system), suggesting that the grammars are capturing very

	In-domain			*Out-of-domain*		
	P	R	F1	P	R	F1
Original	86.79	83.58	85.15	76.04	71.73	73.82
H&S	87.08	83.73	85.37 †	76.33	70.86	73.49
PPDB	86.42	84.64	85.52 †‡	75.37	72.66	73.99 †‡
T3	86.84	84.25	85.52 †	76.04	72.29	74.12
QTSG	87.04	84.34	85.67 †‡	76.88	72.83	74.89 †‡
PPDB+T3	86.61	84.45	85.51 †‡	75.65	72.49	74.03 †‡
PPDB+QTSG	86.70	84.81	85.75 †‡	76.64	73.22	74.89 †‡
T3+QTSG	86.78	84.62	85.69 †‡	76.56	72.88	74.67 †‡
PPDB+QTSG+T3	86.76	84.69	85.71 †‡	76.54	73.19	74.83 †‡
– label projection	80.95	78.75	79.83 †‡	66.94	66.93	66.93 †‡

Table 8: Semantic evaluation results on CoNLL-2009 in-domain and out-of-domain test sets (combining predicate word sense and argument role labels). Results for the models trained on the Original training set, a baseline extension to the training set, extensions due to each grammar and all combinations. – label projection: results from training on the PPDB+QTSG+T3 training corpus, but without rewriting the labels using gold corpus information. † Difference from Original is significant at $p < 0.01$. ‡ Difference from H&S is significant at $p < 0.01$.

		Also produced by this grammar		
		PPDB	T3	QTSG
Proportion of sentences produced by this grammar	H&S	0.4	0.2	28.1
	QTSG	0.0	3.2	

Table 9: Sentence rewrite overlap (%) in the refined rewrite corpora produced by H&S, PPDB, T3 and QTSG.

similar information. For instance, T3 and QTSG are extracted from the same corpora of aligned sentence pairs. The degree of overlap in the rewrite corpora produced by the grammars is shown in Table 9. Although the degree of overlap in exact sentences is low, the relative performance of the resulting models is closer (discussed below). Overall, the best performing system uses transformations obtained from QTSG and PPDB, which is not surprising as the rules extracted from these grammars present minimal overlap.

Benefits also transfer to out-of-domain text for the acquired grammars, improving the overall performance even more than for the in-domain data (see right half in Table 8). The F1-score of the QTSG model is over 1% higher than the original model, and Recall for the model combining all the acquired grammars has increased by 1.5%. Meanwhile, the rewrites of the H&S system do not seem to improve coverage, resulting in a drop in Recall and F1-score.

	In-domain			*Out-of-domain*		
	P	R	F1	P	R	F1
Original	86.79	83.58	85.15	76.04	71.73	73.82
PPDB	87.34	83.10	85.17	76.41	71.42	73.83
T3	87.36	83.26	85.26	76.49	71.76	74.05
QTSG	87.48	83.31	85.34	76.75	72.00	74.30

Table 10: Results on CoNLL-2009 in-domain and out-of-domain test sets, training the SRL model only on rewrites that were labeled as positive.

SVM		Quality		
Thresholds	Count	P	R	F1
None	–	80.26	76.28	78.22
+0.001	10	80.74	76.99	78.82
-0.001	10	80.82	76.87	78.80
-0.2	10	80.65	76.86	78.71
All	10	80.55	77.08	78.78
+0.001	5	80.46	76.54	78.45
+0.001	3	80.00	76.13	78.02

Table 11: Effect of selecting transforms by SVM on the quality of the resulting model (precision, recall and F1 measures on labeling the development set).

In addition, we examined whether filtering the set of acquired transformation functions is indeed beneficial. In the approach we have proposed, transformations are applied to the training corpus twice: the first time as input to an SVM to identify the more reliable rewrite rules, and in a second pass the reduced set of rules is applied to the whole training corpus. An alternative approach would be to apply the transforms only once, and then train the SRL model. We thus took the rewrites that are labeled positive in steps 9–14 of Algorithm 1 and corrected the labels to gold-standard (step 23). SRL models were subsequently trained using the extended training corpus, created by concatenating the original training dataset with these rewrites. Table 10 shows SRL performance for different grammars (PPDB, T3, and QTSG) on the test set. Although precision and F1 have increased over the original model, the gains are much reduced compared to the results obtained using the SVM (Table 8). It appears that the extra rewrites obtained by applying generally-reliable transforms to the whole training set increases coverage, and so improves the performance of the models.

Table 11 shows how altering the quality threshold (and removing indicator features for the number of times each transformation function was extracted) affects performance. Results are shown for the QTSG grammar on the (in-domain) development set (we observed similar patterns for all other grammars and grammar combinations). The SVM quality threshold varied from very positive (no transformations accepted) to very negative (all

	In-domain			*Out-of-domain*		
	P	R	F1	P	R	F1
Original	82.69	78.25	80.41	71.44	65.62	68.40
H&S	83.08	78.45	80.70 †	71.65	64.25	67.75
PPDB	82.34	79.89	81.10 †‡	70.68	67.02	68.80 †‡
T3	82.88	79.30	81.05 †‡	71.54	66.46	68.90 ‡
QTSG	83.00	79.27	81.09 †‡	72.48	66.98	69.62 †‡
PPDB+T3	82.61	79.62	81.09 †‡	71.16	66.88	68.95 †‡
PPDB+QTSG	82.62	80.01	81.29 †‡	72.11	67.47	69.71 †‡
T3+QTSG	82.75	79.77	81.23 †‡	72.08	67.09	69.49 †‡
PPDB+QTSG+T3	82.83	79.95	81.37 †‡	72.08	67.54	69.74 †‡

Table 12: Performance in the labeling of semantic arguments (predicate word sense information removed). † Difference from Original is significant at $p < 0.01$. ‡ Difference from H&S is significant at $p < 0.01$.

	In-domain			*Out-of-domain*		
	P	R	F1	P	R	F1
Original	89.56	84.75	87.09	87.20	80.10	83.50
H&S	89.71	84.71	87.14	87.40	78.38	82.65
PPDB	88.99	86.34	87.65	86.24	81.78	83.95
T3	89.57	85.70	87.59	87.09	80.90	83.88
QTSG	89.53	85.50	87.47	87.28	80.66	83.84
PPDB+T3	89.10	86.28	87.66	86.75	81.53	84.06
PPDB+QTSG	89.28	86.05	87.63	86.77	81.18	83.88
T3+QTSG	89.33	86.10	87.68	87.34	81.29	84.20
PPDB+QTSG+T3	89.33	86.23	87.75	86.90	81.43	84.07

Table 13: Accuracy of identification but not classification (labeling) of semantic arguments.

transformations). These findings indicate that constructing G to be transformations with a positive SVM weight (threshold of +0.001) gives better results that no transformations, or any more permissive threshold.

5.2 Transformation Rules Improve Semantic Role Assignment for Verbal and Nominal Predicates

The results in Table 8 combine accuracy in predicting the sense of predicates and accuracy in labeling their arguments. Generally, the models are better at assigning the correct predicate sense. An interesting result is that much of the gain in performance seen here by rewriting the training corpus comes through improving semantic role assignment. It appears that

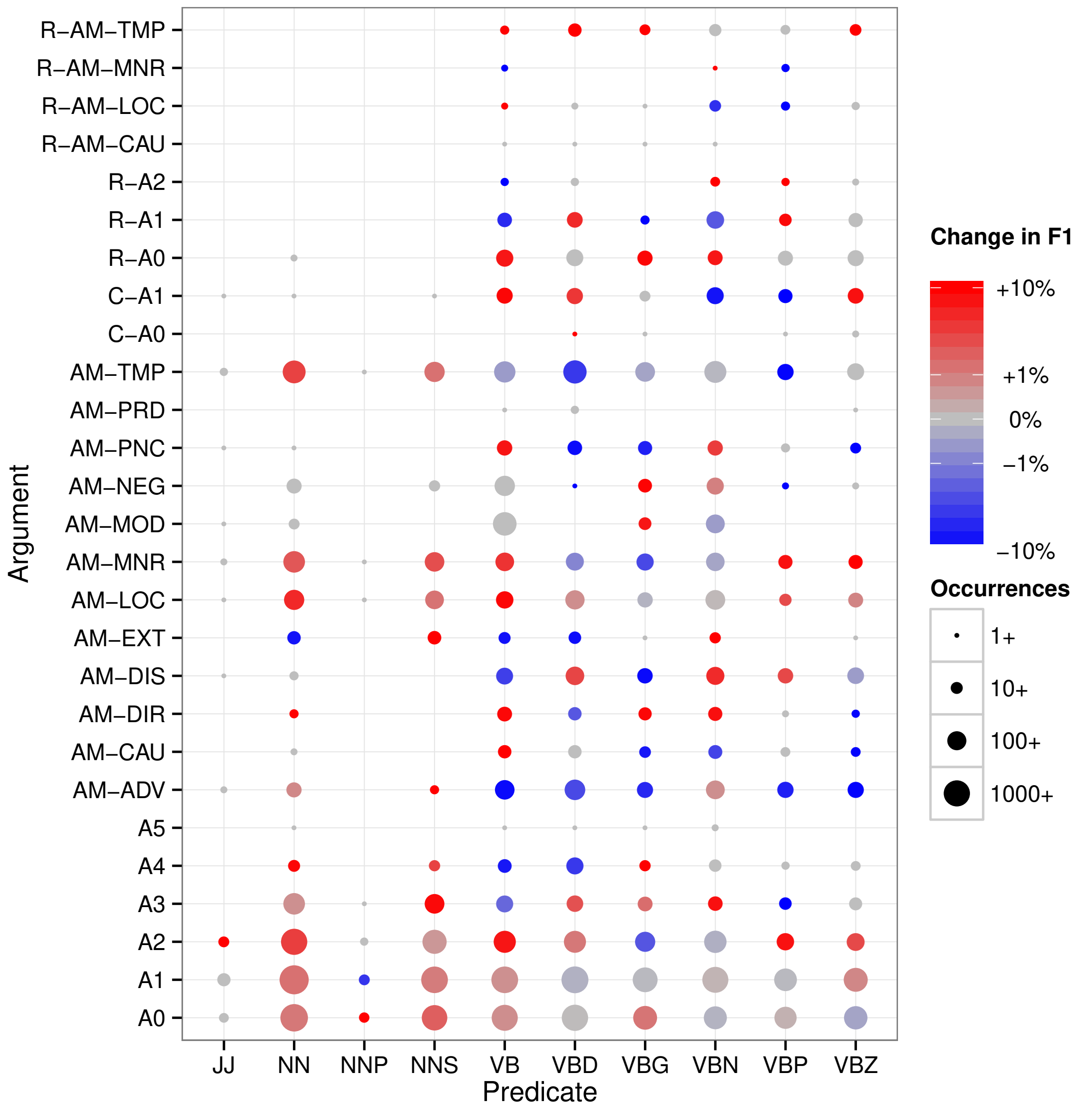

Figure 3: Changes in F1-score for the PPDB+T3+QTSG model over Original, measured by pairs of predicate POS-tag and argument.

introducing syntactic variation in the training data provides the model with wider coverage in syntactic dependency paths between predicate and arguments.

Table 12 shows results for the same models and data sets as above, but focusing on the argument labels only. The acquired grammars show the biggest improvements, with over 1% improvement in Recall in each case, and gains in F1-score between 0.5% and 1.2%. The same models and data sets were used in Table 13, with the results here for argument identification only, not classification (unlabelled arguments). There are improvements over

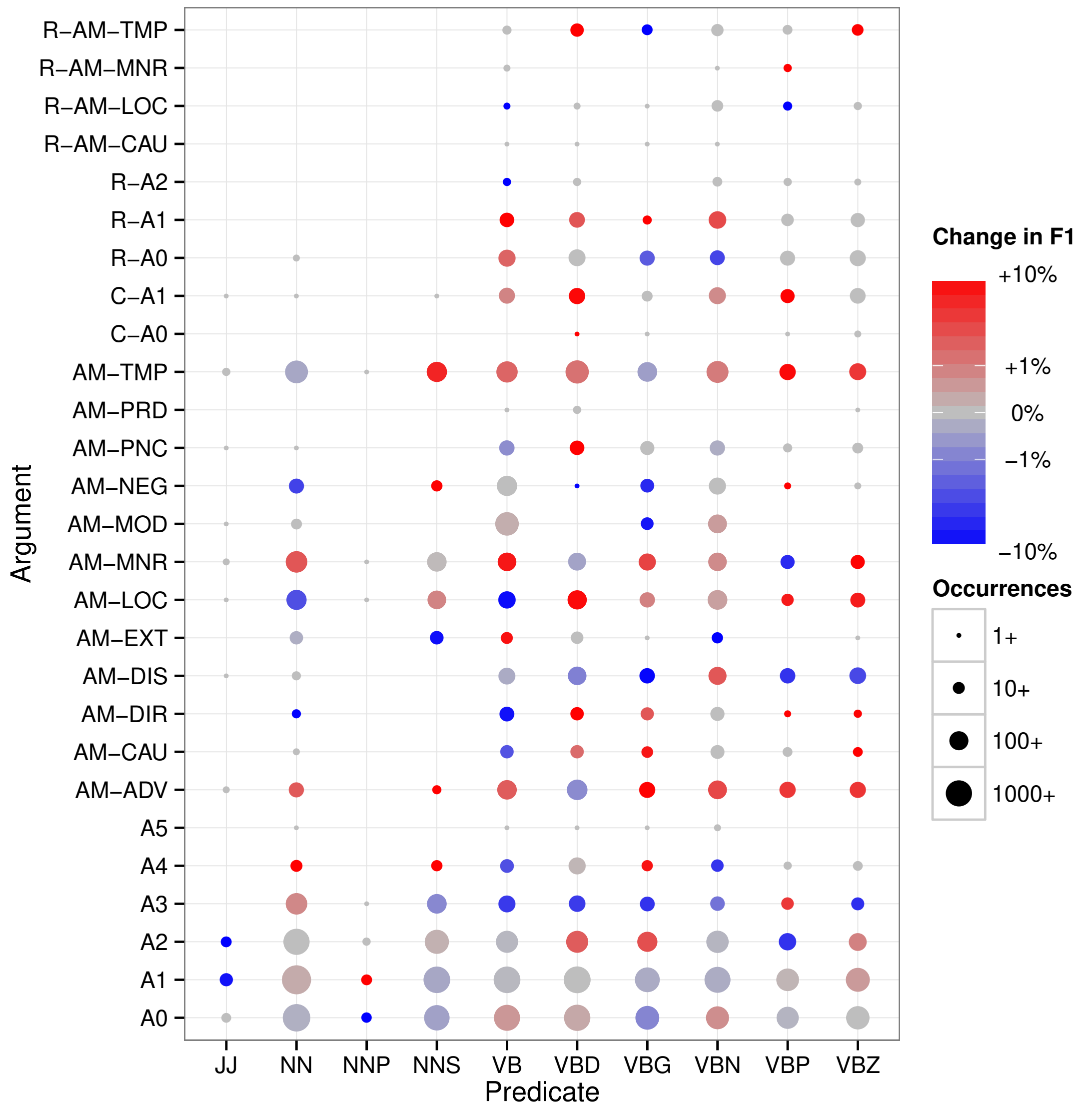

Figure 4: Relative performance in terms of F1-score of the QTSG (red) and PPDB (blue) models, by pairs of predicate POS-tag and argument.

Original in both Recall and F1. They are not as large as before, showing that the overall gains are a result of improvements in both argument identification and classification.

A breakdown of the gains in F1-score by predicate POS-tag and argument is shown in Figure 3, illustrating the relative improvements of the model trained on all acquired grammars (PPDB+T3+QTSG) to the model trained on the original CoNLL training data. This further analysis reveals that most of the gain came from increased precision and recall in predicting the core arguments. There are additional gains in the modifiers of nominal pred-

Dependency path distance	0–1	2	3	4	5	6	7+
Proportion of test set	75.75	13.67	5.54	2.62	1.13	0.56	0.73
SRL model:							
Original	88.83	74.27	61.73	54.76	43.08	23.91	12.27
PPDB	+0.49	+1.43	+2.65	+3.26	+4.78	+5.53	–0.06
T3	+0.63	+1.15	+1.65	+1.67	+1.42	+1.22	+0.69
QTSG	+0.53	+0.66	+1.82	+2.98	+3.01	–0.96	+0.43
PPDB+QTSG+T3	+0.74	+1.60	+2.49	+2.02	+6.20	+1.35	+1.61

Table 14: F1-scores for labeled arguments where distance between predicate and argument is measured as the number of arcs in the dependency graph. Results are from the CoNLL in-domain test set. Lower rows show the change in F1-score over the Original SRL model.

icates. There was some improvement and some losses in the very common core arguments (A0 and A1) of the verbal predicates, but the more striking gains were seen for the other core argument labels. This seems consistent with the models learning from wider syntactic coverage. Figure 4 shows a similar breakdown of the gains in F1-score by predicate POS-tag and argument, this time comparing the improvements seen from the QTSG corpus with those resulting from PPDB. The differences are less pronounced, with PPDB improving the core arguments more, and QTSG improving performance in labeling modifiers.

We also investigated the effect of the label projection mechanism itself. We used the rewrites produced by all grammars (PPDB+T3+QTSG) to extend the training set. However, instead of using projected labels, we used the the original model M_{gold} (trained on the training partition of CoNLL-2009) to label the refined corpus. We then retrained on the extended corpus and used this retrained model to label the test corpus. In other words, we removed step 25 in Algorithm 1. This can be considered as a form of self-training. Results on both the test and out-of-domain sets show that using automatically generated labels instead of projected ones seriously impairs the resulting model, with F1-scores decreasing by almost 6% on the test set and 8% on the out-of-domain set (see last row of Table 8).

5.3 Transformation Rules Improve Performance of Relations Involving Long Dependency Paths

The *dependency path* (the sequence of arcs through the syntactic dependency tree) between a predicate and its argument is typically short. Table 14 shows that in the gold-labeled test set, three-quarters of the arguments are direct dependency heads or children of the predicate, or in the case of nominal predicates, the argument is the predicate itself. Existing SRL models are highly accurate over these shorter paths—the original SRL model has an F1-score of almost 89%—but prediction accuracy drops considerably as the dependency path grows. As can be seen in Table 14, adding rewrites to the training set improves prediction accuracy for almost all combinations of transformation grammar and dependency path distance, and the largest gains are seen when the number of arcs in the dependency path

	In-domain			*Out-of-domain*		
	P	R	F1	P	R	F1
Original	88.44	84.42	86.38	77.89	72.73	75.22
H&S	88.68	84.34	86.46	78.11	71.76	74.80
PPDB	86.42	84.64	85.52 †‡	76.73	73.36	75.01 †‡
T3	88.04	84.78	86.38	77.07	72.97	74.97
QTSG	88.41	85.05	86.70 †‡	78.34	73.70	75.95 ‡
T3+QTSG	88.24	85.21	86.70 †‡	78.00	73.53	75.70 ‡
PPDB+T3	86.61	84.45	85.51 †‡	77.30	73.51	75.35 †‡
PPDB+QTSG	86.70	84.81	85.75 †‡	77.41	73.82	75.57 †‡
PPDB+QTSG+T3	87.94	85.25	86.57 †‡	77.67	73.73	75.64 †‡

Table 15: Results on the CoNLL test sets for models combining extended training data and global reranker. † Difference from Original is significant at $p < 0.01$. ‡ Difference from H&S is significant at $p < 0.01$.

is between three and six. Improvements in F1-score are observed for individual grammars and their combination (PPDB+QTSG+T3).

5.4 Transformation Rules Improve Performance Even When a Global Reranker is Used

The SRL system we used (Björkelund et al., 2009) can optionally incorporate a global reranker (Toutanova, Haghighi, & Manning, 2005). The reranker re-scores the complete predicate-argument structure, using features from all stages of the local pipeline and additional features representing the sequence of core argument labels for the current predicate. Table 15 presents evaluation results for a global reranker trained with the extended corpora produced by our method. Compared to the model trained on the original corpus, adding the reranker does provide significant improvement.[3] Training on the extended data gives further increases in performance; these are now smaller, though still significant, than was the case in Table 8. This indicates that the global reranker is compensating for some, but not all, of the new information contained in the extended training data.

5.5 Transformation Rules Improve Performance Across (Small and Large) Datasets

We also investigated the accuracy of the labeler as a function of the size of the original training data. For each size, subsets of the original training data were created (with replacement) and used to train the SRL model, and the performance of each resulting model measured using the development set. For each training subset, we applied Algorithm 1: the original SRL model was trained only on the subset; we created an extended corpus from

3. The scores reported here are higher than the official CoNLL-2009 ones (in domain P:87.46, R:83.87, F1:85.63; out of domain P:76.04, R:70.76, F1:73.31) through using the MATE-TOOLS NLP pipeline for the dependency parse, rather than the dependency information provided in the test set.

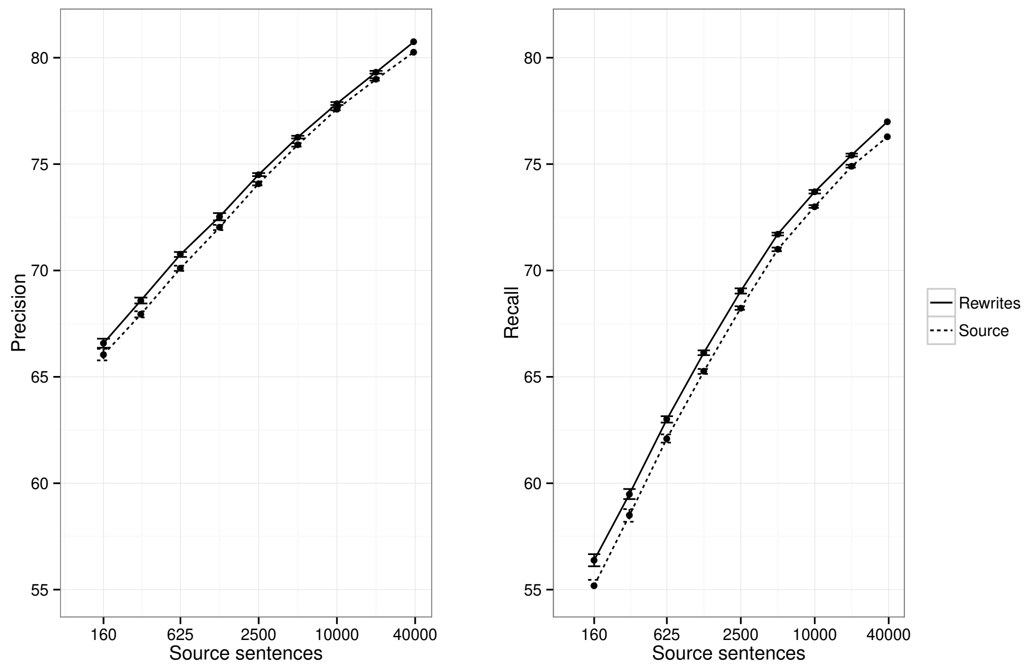

Figure 5: SRL model performance as a function of the size of the training data, with and without additional rewrites. Error bars show standard error over 10 experiments.

the subset using the grammar; an SVM was trained each time to refine the transformations to those that preserved labels; and the SRL model retrained on the original plus refined rewritten version of the corpus subset.

In particular, we wanted to investigate if the rewritten text provided a performance benefit when there was only a small amount of training data, and any such benefit would be subsumed if more labeled training data was provided. The learning curves in Figure 5 show the contrary: while increasing the quantity of source training data undoubtedly improves the quality of the SRL model, we found that including the rewritten training data in addition consistently improves both precision and recall measures. The learning curves in Figure 5 use the QTSG grammar as the set of transformation functions; we obtained similar results with PPDB and T3 (and all grammar combinations), however we omit them for the sake of brevity.

6. Conclusions

In this paper we investigated the potential of text rewriting as a means of increasing the amount of training data available for supervised NLP tasks. Our method automatically extracts rewrite rules from comparable corpora and uses them to generate multiple syntactic variants for sentences annotated with gold standard labels. Application of our method to semantic role labeling reveals that syntactic transformations improve SRL performance

	$\langle$NP, NP$\rangle$	$\rightarrow$	$\langle$[NP DT$_{\boxed{1}}$ JJ$_{\boxed{\epsilon}}$ NNS$_{\boxed{2}}$], [NP DT$_{\boxed{1}}$ NNS$_{\boxed{2}}$]$\rangle$
QTSG	$\langle$NP, NP$\rangle$	$\rightarrow$	$\langle$[NP NP$_{\boxed{1}}$, NP$_{\boxed{\epsilon}}$ CC NP$_{\boxed{\epsilon}}$], [NP NP$_{\boxed{1}}$]$\rangle$
	$\langle$NP, S$\rangle$	$\rightarrow$	$\langle$[NP NP$_{\boxed{1}}$ PP$_{\boxed{2}}$], [S *It is* NP$_{\boxed{1}}$ PP$_{\boxed{2}}$.]$\rangle$
	$\langle\bar{S}, \bar{S}\rangle$	$\rightarrow$	$\langle$[$\bar{S}$ *even if it* VBZ$_{\boxed{1}}$ NP$_{\boxed{2}}$], [$\bar{S}$ *even though it* VBZ$_{\boxed{1}}$ NP$_{\boxed{2}}$]$\rangle$
PPDB	$\langle$ADJP, ADJP$\rangle$	$\rightarrow$	$\langle$[ADJP *just as* JJ$_{\boxed{1}}$], [ADJP *equally* JJ$_{\boxed{1}}$]$\rangle$
	$\langle$PP, PP$\rangle$	$\rightarrow$	$\langle$[PP *in the past month*], [PP *in the last month*]$\rangle$

Table 16: Examples of QTSG and PPDB synchronous grammar rules given high importance during refinement. Boxed indices indicate alignment, $\sim$.

beyond the sate of the art on the CoNLL 2009 benchmark dataset. Specifically, we experimentally show that (a) rewrite rules, whether automatic or hand-written, consistently improve SRL performance, although automatic variants tend to perform best; (b) syntactic transformations improve SRL performance both within- and out-of-domain; and (c) improvements are observed across learners, even when using a global reranker.

In the future we would like to explore better ways of identifying the best (i.e., performance enhancing) rewrite rules which may be task and grammar specific. Table 16 illustrates the rules deemed important (i.e., given high weight) by our SVM classifier for the SRL task. For instance, we could undertake more detailed feature engineering, including tree-based and ngram features to capture the grammaticality of the rewritten sentences. Throughout this paper we have argued that transformation rules can be used to enhance performance in the SRL task. Conversely, some of the work described here might be of relevance to other NLP tasks employing rewriting. For example, the idea of identifying label preserving transformations, could be used to learn which rules are meaning preserving and consequently safe to use in tasks such simplification or sentence compression. Machine translation, textual entailment, and semantic parsing are additional application areas which stand to benefit from more accurate rewrite rules. Much of the methodology reported here could be adapted to machine translation either for training with larger datasets (Callison-Burch, Koehn, & Osborne, 2006), for domain-adaptation (Irvine, Quirk, & Daumé III, 2013), or evaluation (Kauchak & Barzilay, 2006)

Finally, beyond supervised SRL, we would like to adapt our method to unsupervised semantic role induction (Lang & Lapata, 2011; Titov & Klementiev, 2012), investigate alternative synchronous grammar extraction methods (e.g., based on dependency information), and obtain rewrite rules from larger comparable corpora.

Acknowledgments

We are grateful to the anonymous referees whose feedback helped to substantially improve the present paper. We acknowledge the financial support of EPSRC (EP/K017845/1) in the framework of the CHIST-ERA READERS project.

References

Aho, A. V., & Ullman, J. D. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, *3*(1), 37–56.

Bannard, C., & Callison-Burch, C. (2005a). Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 597–604, Ann Arbor.

Bannard, C., & Callison-Burch, C. (2005b). Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of the 43rd ACL*, pp. 255–262, Ann Arbor, MI.

Barzilay, R., & McKeown, K. (2001). Extracting Paraphrases from a Parallel Corpus. In *Proceedings of the ACL/EACL*, pp. 50–57, Toulouse, France.

Björkelund, A., Hafdell, L., & Nugues, P. (2009). Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pp. 43–48, Boulder, Colorado. Software retrieved from `https://code.google.com/p/mate-tools/`.

Callison-Burch, C. (2007). *Paraphrasing and Translation*. Ph.D. thesis, University of Edinburgh.

Callison-Burch, C. (2008). Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pp. 196–205, Honolulu, Hawaii.

Callison-Burch, C., Koehn, P., & Osborne, M. (2006). Improved statistical machine translation using paraphrases. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 17–24, New York City, USA.

Chandrasekar, R., Doran, C., & Srinivas, B. (1996). Motivations and Methods for Text Simplification. In *Proceedings of the 16th International Conference on Computational Linguistics*, pp. 1041–1044, Copenhagen, Denmark.

Chiang, D. (2007). Hierarchical Phrase-Based Translation. *Computational Linguistics*, *33*(2), 201–228.

Cohn, T., & Lapata, M. (2009). Sentence Compression as Tree Transduction. *Journal of Artificial Intelligence Research*, *34*, 637–674.

Cohn, T., & Lapata, M. (2013). An abstractive approach to sentence compression. *ACM Trans. Intell. Syst. Technol.*, *4*(3), 41:1–41:35.

Coster, W., & Kauchak, D. (2011). Simple English Wikipedia: A New Text Simplification Task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 665–669, Portland, Oregon, USA.

Dowty, D. (1991). Thematic Proto Roles and Argument Selection. *Language*, *67*(3), 547–619.

Eisner, J. (2003). Learning Non-Isomorphic Tree Mappings for Machine Translation. In *Proceedings of the ACL Interactive Poster/Demonstration Sessions*, pp. 205–208, Sapporo, Japan.

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Feblowitz, D., & Kauchak, D. (2013). Sentence simplification as tree transduction. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pp. 1–10, Sofia, Bulgaria.

Fürstenau, H., & Lapata, M. (2012). Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, *38*(1), 135–171.

Galley, M., & McKeown, K. (2007). Lexicalized Markov Grammars for Sentence Compression. In *Proceedings of the NAACL/HLT*, pp. 180–187, Rochester, NY.

Ganitkevitch, J., Callison-Burch, C., Napoles, C., & Van Durme, B. (2011). Learning Sentential Paraphrases from Bilingual Parallel Corpora for Text-to-Text Generation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 1168–1179, Edinburgh, Scotland, UK.

Ganitkevitch, J., Cao, Y., Weese, J., Post, M., & Callison-Burch, C. (2012). Joshua 4.0: Packing, pro, and paraphrases. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pp. 283–291, Montréal, Canada.

Ganitkevitch, J., Van Durme, B., & Callison-Burch, C. (2013). PPDB: The Paraphrase Database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 758–764, Atlanta, Georgia. We used the prepackaged "small" constituent syntactic subset of PPDB, retrieved from `http://paraphrase.org`.

Gildea, D., & Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics*, *28*(3), 245–288.

Graehl, J., & Knight, K. (2004). Training Tree Transducers. In *HLT-NAACL 2004: Main Proceedings*, pp. 105–112, Boston, MA.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., & Zhang, Y. (2009). The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pp. 1–18, Boulder, Colorado.

Heilman, M., & Smith, N. (2010). Extracting Simplified Statements for Factual Question Generation. In *Proceedings of the 3rd Workshop on Question Generation*, pp. 11–20, Carnegie Mellon University, PA. Software available at `http://www.ark.cs.cmu.edu/mheilman/questions/`.

Irvine, A., Quirk, C., & Daumé III, H. (2013). Monolingual marginal matching for translation model adaptation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1077–1088, Seattle, Washington, USA.

Kauchak, D. (2013). Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1537–1546, Sofia, Bulgaria. We used the *Version 2.0 sentence-aligned* corpus, retrieved from `http://www.cs.middlebury.edu/~dkauchak/simplification/`.

Kauchak, D., & Barzilay, R. (2006). Paraphrasing for automatic evaluation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 455–462, New York City, USA.

Klebanov, B. B., Knight, K., & Marcu, D. (2004). Text Simplification for Information-Seeking Applications. In Meersman, R., & Tari, Z. (Eds.), *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pp. 735–747. Springer Berlin Heidelberg.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press.

Koehn, P., Och, F. J., & Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the HLT/NAACL*, pp. 48–54, Edmonton, Canada.

Kundu, G., & Roth, D. (2011). Adapting Text instead of the Model: An Open Domain Approach. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pp. 229–237, Portland, Oregon, USA.

Kwiatkowski, T. (2012). *Probabilistic Grammar Induction from Sentences and Structured Meanings*. Ph.D. thesis, University of Edinburgh.

Lang, J., & Lapata, M. (2011). Unsupervised semantic role induction via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1117–1126, Portland, Oregon, USA.

Liang, P., Taskar, B., & Klein, D. (2006). Alignment by Agreement. In *Proceedings of the HLT/NAACL*, pp. 104–111, New York, NY.

Marton, Y., Callison-Burch, C., & Resnik, P. (2009). Improved statistical machine translation using monolingually-derived paraphrases. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 381–390, Singapore.

Mehdad, Y., Negri, M., & Federico, M. (2010). Towards cross-lingual textual entailment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 321–324, Los Angeles, California.

Melli, G., Wang, Y., Liu, Y., Kashani, M. M., Shi, Z., Gu, B., Sarkar, A., & Popowich, F. (2005). Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2005 Summarization Task. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing Document Understanding Workshop*, Vancouver, Canada.

Noreen, E. (1989). *Computer-intensive methods for testing hypotheses: an introduction*. Wiley.

Padó, S. (2006). *User's guide to `sigf`: Significance testing by approximate randomisation*. Retrieved from `http://www.nlpado.de/~sebastian/software/sigf.shtml`.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, *31*(1), 71–106.

Pang, B., Knight, K., & Marcu, D. (2003). Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *Proceedings of the NAACL*, pp. 181–188, Edmonton, Canada.

Park, J. H., Croft, B., & Smith, D. A. (2011). A Quasi-synchronous Dependence Model for Information Retrieval. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 17–26, Glasgow, United Kingdom.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Shen, D., & Lapata, M. (2007). Using Semantic Roles to Improve Question Answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 12–21, Prague, Czech Republic.

Surdeanu, M., Harabagiu, S., Williams, J., & Aarseth, P. (2003). Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 8–15, Sapporo, Japan.

Titov, I., & Klementiev, A. (2012). A bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 12–22, Avignon, France.

Toutanova, K., Haghighi, A., & Manning, C. (2005). Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 589–596, Ann Arbor, Michigan.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc.

Vickrey, D., & Koller, D. (2008). Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pp. 344–352, Columbus, Ohio.

Wang, M., & Manning, C. (2010). Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pp. 1164–1172, Beijing, China.

Wang, M., Smith, N. A., & Mitamura, T. (2007). What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 22–32, Prague, Czech Republic.

Woodsend, K., & Lapata, M. (2011). Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 409–420, Edinburgh, Scotland, UK. We used the *Wikipedia revisions* corpus, retrieved from `http://homepages.inf.ed.ac.uk/kwoodsen/wiki.html`.

Woodsend, K., & Lapata, M. (2012). Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods* *in Natural Language Processing and Computational Natural Language Learning*, pp. 233–243, Jeju Island, Korea.

Wu, D. (1997). Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, *23*(3), 377–404.

Wu, D., & Fung, P. (2009). Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pp. 13–16, Boulder, Colorado.

Yamada, K., & Knight, K. (2001). A Syntax-based Statistical Translation Model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pp. 523–530, Toulouse, France.

Yamangil, E., & Nelken, R. (2008). Mining Wikipedia revision histories for improving sentence compression. In *Proceedings of ACL-08: HLT, Short Papers*, pp. 137–140, Columbus, Ohio.

Zanzotto, F. M., & Pennacchiotti, M. (2010). Expanding textual entailment corpora fromwikipedia using co-training. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pp. 28–36, Beijing, China. Coling 2010 Organizing Committee.

Zhu, Z., Bernhard, D., & Gurevych, I. (2010). A Monolingual Tree-based Translation Model for Sentence Simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 1353–1361, Beijing, China.