Paraphrase Identification as Probabilistic Quasi-Synchronous Recognition

Dipanjan Das and **Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{dipanjan,nasmith}@cs.cmu.edu

Abstract

We present a novel approach to deciding whether two sentences hold a paraphrase relationship. We employ a generative model that generates a paraphrase of a given sentence, and we use probabilistic inference to reason about whether two sentences share the paraphrase relationship. The model cleanly incorporates both syntax and lexical semantics using quasi-synchronous dependency grammars (Smith and Eisner, 2006). Furthermore, using a product of experts (Hinton, 2002), we combine the model with a complementary logistic regression model based on state-of-the-art lexical overlap features. We evaluate our models on the task of distinguishing true paraphrase pairs from false ones on a standard corpus, giving competitive state-of-the-art performance.

1 Introduction

The problem of modeling *paraphrase* relationships between natural language utterances (McKeown, 1979) has recently attracted interest. For computational linguists, solving this problem may shed light on how best to model the semantics of sentences. For natural language engineers, the problem bears on information management systems like abstractive summarizers that must measure semantic overlap between sentences (Barzilay and Lee, 2003), question answering modules (Marsi and Krahmer, 2005) and machine translation (Callison-Burch et al., 2006).

The paraphrase identification problem asks whether two sentences have essentially the same meaning. Although paraphrase identification is defined in semantic terms, it is usually solved using statistical classifiers based on shallow lexical, n-gram, and syntactic "overlap" features. Such overlap features give the best-published classification accuracy for the paraphrase identification task (Zhang and Patrick, 2005; Finch et al., 2005; Wan et al., 2006; Corley and Mihalcea, 2005, *inter alia*), but do not explicitly model correspondence structure (or "alignment") between the parts of two sentences. In this paper, we adopt a model that posits correspondence between the words in the two sentences, defining it in *loose syntactic* terms: if two sentences are paraphrases, we expect their dependency trees to align closely, though some divergences are also expected, with some more likely than others. Following Smith and Eisner (2006), we adopt the view that the syntactic structure of sentences paraphrasing some sentence $\mathbf{s}$ should be "inspired" by the structure of $\mathbf{s}$.

Because dependency syntax is still only a crude approximation to *semantic* structure, we augment the model with a lexical semantics component, based on WordNet (Miller, 1995), that models how *words* are probabilistically altered in generating a paraphrase. This combination of loose syntax and lexical semantics is similar to the "Jeopardy" model of Wang et al. (2007).

This syntactic framework represents a major departure from useful and popular surface similarity features, and the latter are difficult to incorporate into our probabilistic model. We use a product of experts (Hinton, 2002) to bring together a logistic regression classifier built from n-gram overlap features and our syntactic model. This combined model leverages complementary strengths of the two approaches, outperforming a strong state-of-the-art baseline (Wan et al., 2006).

This paper is organized as follows. We introduce our probabilistic model in §2. The model makes use of three quasi-synchronous grammar models (Smith and Eisner, 2006, QG, hereafter) as components (one modeling paraphrase, one modeling not-paraphrase, and one a base grammar); these are detailed, along with latent-variable inference and discriminative training algorithms, in §3. We discuss the Microsoft Research Paraphrase Corpus, upon which we conduct experiments, in §4. In §5, we present experiments on paraphrase

identification with our model and make comparisons with the existing state-of-the-art. We describe the product of experts and our lexical overlap model, and discuss the results achieved in §6. We relate our approach to prior work (§7) and conclude (§8).

2 Probabilistic Model

Since our task is a classification problem, we require our model to provide an estimate of the posterior probability of the relationship (i.e., "paraphrase," denoted p, or "not paraphrase," denoted n), given the pair of sentences.[1] Here, p_Q denotes model probabilities, c is a relationship class (p or n), and $\mathbf{s}_1$ and $\mathbf{s}_2$ are the two sentences. We choose the class according to:

$$\begin{aligned}\hat{c} &= \operatorname*{argmax}_{c \in \{\mathsf{p},\mathsf{n}\}} p_Q(c \mid \mathbf{s}_1, \mathbf{s}_2) \\ &= \operatorname*{argmax}_{c \in \{\mathsf{p},\mathsf{n}\}} p_Q(c) \times p_Q(\mathbf{s}_1, \mathbf{s}_2 \mid c) \quad (1)\end{aligned}$$

We define the class-conditional probabilities of the two sentences using the following generative story. First, grammar G_0 generates a sentence $\mathbf{s}$. Then a class c is chosen, corresponding to a class-specific *probabilistic quasi-synchronous grammar* G_c. (We will discuss QG in detail in §3. For the present, consider it a specially-defined probabilistic model that generates sentences with a specific property, like "paraphrases $\mathbf{s}$," when $c = \mathsf{p}$.) Given $\mathbf{s}$, G_c generates the other sentence in the pair, $\mathbf{s}'$.

When we observe a pair of sentences $\mathbf{s}_1$ and $\mathbf{s}_2$ we do not presume to know which came first (i.e., which was $\mathbf{s}$ and which was $\mathbf{s}'$). Both orderings are assumed to be equally probable. For class c,

$$\begin{aligned}p_Q(\mathbf{s}_1, \mathbf{s}_2 \mid c) = \\ 0.5 \times p_Q(\mathbf{s}_1 \mid G_0) \times p_Q(\mathbf{s}_2 \mid G_c(\mathbf{s}_1)) \\ + \quad 0.5 \times p_Q(\mathbf{s}_2 \mid G_0) \times p_Q(\mathbf{s}_1 \mid G_c(\mathbf{s}_2)) \quad (2)\end{aligned}$$

where c can be p or n; $G_{\mathsf{p}}(\mathbf{s})$ is the QG that generates paraphrases for sentence $\mathbf{s}$, while $G_{\mathsf{n}}(\mathbf{s})$ is the QG that generates sentences that are *not* paraphrases of sentence $\mathbf{s}$. This latter model may seem counter-intuitive: since the vast majority of possible sentences are not paraphrases of $\mathbf{s}$, why is a special grammar required? Our use of a G_{n} follows from the properties of the corpus currently used for learning, in which the negative examples were selected to have high lexical overlap. We return to this point in §4.

[1] Although we do not explore the idea here, the model could be adapted for other sentence-pair relationships like entailment or contradiction.

3 QG for Paraphrase Modeling

Here, we turn to the models G_{p} and G_{n} in detail.

3.1 Background

Smith and Eisner (2006) introduced the quasi-synchronous grammar formalism. Here, we describe some of its salient aspects. The model arose out of the empirical observation that translated sentences have *some* isomorphic syntactic structure, but divergences are possible. Therefore, rather than an isomorphic structure over a pair of source and target sentences, the syntactic tree over a target sentence is modeled by a source sentence-specific grammar "inspired" by the source sentence's tree. This is implemented by associating with each node in the target tree a subset of the nodes in the source tree. Since it loosely links the two sentences' syntactic structures, QG is well suited for problems like word alignment for MT (Smith and Eisner, 2006) and question answering (Wang et al., 2007).

Consider a very simple quasi-synchronous context-free dependency grammar that generates one dependent per production rule.[2] Let $\mathbf{s} = \langle s_1, ..., s_m \rangle$ be the source sentence. The grammar rules will take one of the two forms:

$$\langle t, l \rangle \rightarrow \langle t, l \rangle \langle t', k \rangle \text{ or } \langle t, l \rangle \rightarrow \langle t', k \rangle \langle t, l \rangle$$

where t and t' range over the vocabulary of the target language, and l and $k \in \{0, ..., m\}$ are indices in the source sentence, with 0 denoting null.[3] Hard or soft constraints can be applied between l and k in a rule. These constraints imply permissible "configurations." For example, requiring $l \neq 0$ and, if $k \neq 0$ then s_k must be a child of s_l in the source tree, we can implement a synchronous dependency grammar similar to (Melamed, 2004).

Smith and Eisner (2006) used a quasi-synchronous grammar to *discover* the correspondence between words implied by the correspondence between the trees. We follow Wang et al. (2007) in treating the correspondences as latent variables, and in using a WordNet-based lexical semantics model to generate the target words.

[2] Our actual model is more complicated; see §3.2.

[3] A more general QG could allow one-to-many alignments, replacing l and k with *sets* of indices.

3.2 Detailed Model

We describe how we model $p_Q(\mathbf{t} \mid G_{\mathsf{p}}(\mathbf{s}))$ and $p_Q(\mathbf{t} \mid G_{\mathsf{n}}(\mathbf{s}))$ for source and target sentences $\mathbf{s}$ and $\mathbf{t}$ (appearing in Eq. 2 alternately as $\mathbf{s}_1$ and $\mathbf{s}_2$).

A dependency tree on a sequence $\mathbf{w} = \langle w_1, ..., w_k \rangle$ is a mapping of indices of words to indices of syntactic parents, $\tau_p : \{1, ..., k\} \rightarrow \{0, ..., k\}$, and a mapping of indices of words to dependency relation types in $\mathcal{L}$, $\tau_\ell : \{1, ..., k\} \rightarrow \mathcal{L}$. The set of indices children of w_i to its left, $\{j : \tau^{\mathbf{w}}(j) = i, j < i\}$, is denoted $\lambda^{\mathbf{w}}(i)$, and $\rho^{\mathbf{w}}(i)$ is used for right children. w_i has a single parent, denoted by $w_{\tau_p(i)}$. Cycles are not allowed, and w_0 is taken to be the dummy "wall" symbol, \$, whose only child is the root word of the sentence (normally the main verb). The label for w_i is denoted by $\tau_\ell(i)$. We denote the whole tree of a sentence $\mathbf{w}$ by $\tau^{\mathbf{w}}$, the subtree rooted at the ith word by $\tau^{\mathbf{w},i}$.

Consider two sentences: let the source sentence $\mathbf{s}$ contain m words and the target sentence $\mathbf{t}$ contain n words. Let the correspondence $x : \{1, ..., n\} \rightarrow \{0, ..., m\}$ be a mapping from indices of words in $\mathbf{t}$ to indices of words in $\mathbf{s}$. (We require each target word to map to at most one source word, though multiple target words can map to the same source word, i.e., $x(i) = x(j)$ while $i \neq j$.) When $x(i) = 0$, the ith target word maps to the wall symbol, equivalently a "null" word. Each of our QGs G_{p} and G_{n} generates the alignments x, the target tree $\tau^{\mathbf{t}}$, and the sentence $\mathbf{t}$. Both G_{p} and G_{n} are structured in the same way, differing only in their parameters; henceforth we discuss G_{p}; G_{n} is similar.

We assume that the parse trees of $\mathbf{s}$ and $\mathbf{t}$ are known.[4] Therefore our model defines:

$$\begin{aligned} p_Q(\mathbf{t} \mid G_{\mathsf{p}}(\mathbf{s})) &= p(\tau^{\mathbf{t}} \mid G_{\mathsf{p}}(\tau^{\mathbf{s}})) \\ &= \textstyle\sum_x p(\tau^{\mathbf{t}}, x \mid G_{\mathsf{p}}(\tau^{\mathbf{s}})) \quad (3) \end{aligned}$$

Because the QG is essentially a context-free dependency grammar, we can factor it into recursive steps as follows (let i be an arbitrary index in $\{1, ..., n\}$):

$$P(\tau^{\mathbf{t},i} \mid t_i, x(i), \tau^{\mathbf{s}}) = p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid t_i)$$
$$\times \prod_{j \in \lambda^{\mathbf{t}}(i) \cup \rho^{\mathbf{t}}(i)} \sum_{x(j)=0}^{m} P(\tau^{\mathbf{t},j} \mid t_j, x(j), \tau^{\mathbf{s}})$$
$$\times p_{kid}(t_j, \tau^{\mathbf{t}}_\ell(j), x(j) \mid t_i, x(i), \tau^{\mathbf{s}}) \quad (4)$$

where p_{val} and p_{kid} are valence and child-production probabilities parameterized as discussed in §3.4. Note the recursion in the second-to-last line.

We next describe a dynamic programming solution for calculating $p(\tau^{\mathbf{t}} \mid G_{\mathsf{p}}(\tau^{\mathbf{s}}))$. In §3.4 we discuss the parameterization of the model.

3.3 Dynamic Programming

Let $C(i, l)$ refer to the probability of $\tau^{\mathbf{t},i}$, assuming that the parent of t_i, $t_{\tau^{\mathbf{t}}_p(i)}$, is aligned to s_l. For leaves of $\tau^{\mathbf{t}}$, the base case is:

$$\begin{aligned} C(i, l) &= p_{val}(0, 0 \mid t_i) \times \quad (5) \\ &\quad \textstyle\sum_{k=0}^{m} p_{kid}(t_i, \tau^{\mathbf{t}}_\ell(i), k \mid t_{\tau^{\mathbf{t}}_p(i)}, l, \tau^{\mathbf{s}}) \end{aligned}$$

where k ranges over possible values of $x(i)$, the source-tree node to which t_i is aligned. The recursive case is:

$$\begin{aligned} C(i, l) &= p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid t_i) \quad (6) \\ &\quad \times \textstyle\sum_{k=0}^{m} p_{kid}(t_i, \tau^{\mathbf{t}}_\ell(i), k \mid t_{\tau^{\mathbf{t}}_p(i)}, l, \tau^{\mathbf{s}}) \\ &\quad \times \textstyle\prod_{j \in \lambda^{\mathbf{t}}(i) \cup \rho^{\mathbf{t}}(i)} C(j, k) \end{aligned}$$

We assume that the wall symbols t_0 and s_0 are aligned, so $p(\tau^{\mathbf{t}} \mid G_p(\tau^{\mathbf{s}})) = C(r, 0)$, where r is the index of the root word of the target tree $\tau^{\mathbf{t}}$. It is straightforward to show that this algorithm requires $O(m^2 n)$ runtime and $O(mn)$ space.

3.4 Parameterization

The valency distribution p_{val} in Eq. 4 is estimated in our model using the transformed treebank (see footnote 4). For unobserved cases, the conditional probability is estimated by backing off to the parent POS tag and child direction.

We discuss next how to parameterize the probability p_{kid} that appears in Equations 4, 5, and 6. This conditional distribution forms the core of our QGs, and we deviate from earlier research using QGs in defining p_{kid} in a fully generative way.

In addition to assuming that dependency parse trees for $\mathbf{s}$ and $\mathbf{t}$ are observable, we also assume each word w_i comes with POS and named entity tags. In our experiments these were obtained automatically using MXPOST (Ratnaparkhi, 1996) and BBN's Identifinder (Bikel et al., 1999).

[4] In our experiments, we use the parser described by McDonald et al. (2005), trained on sections 2–21 of the *WSJ* Penn Treebank, transformed to dependency trees following Yamada and Matsumoto (2003). (The same treebank data were also to estimate many of the parameters of our model, as discussed in the text.) Though it leads to a partial "pipeline" approximation of the posterior probability $p(c \mid \mathbf{s}, \mathbf{t})$, we believe that the relatively high quality of English dependency parsing makes this approximation reasonable.

For clarity, let $j = \tau_p^{\mathbf{t}}(i)$ and let $l = x(j)$.

$$p_{kid}(t_i, \tau_\ell^{\mathbf{t}}(i), x(i) \mid t_j, l, \tau^{\mathbf{s}}) =$$
$$p_{config}(config(t_i, t_j, s_{x(i)}, s_l) \mid t_j, l, \tau^{\mathbf{s}}) \quad (7)$$
$$\times p_{unif}(x(i) \mid config(t_i, t_j, s_{x(i)}, s_l)) \quad (8)$$
$$\times p_{lab}(\tau_\ell^{\mathbf{t}}(i) \mid config(t_i, t_j, s_{x(i)}, s_l)) \quad (9)$$
$$\times p_{pos}(pos(t_i) \mid pos(s_{x(i)})) \quad (10)$$
$$\times p_{ne}(ne(t_i) \mid ne(s_{x(i)})) \quad (11)$$
$$\times p_{lsrel}(lsrel(t_i) \mid s_{x(i)}) \quad (12)$$
$$\times p_{word}(t_i \mid lsrel(t_i), s_{x(i)}) \quad (13)$$

We consider each of the factors above in turn.

Configuration In QG, "configurations" refer to the tree relationship among source-tree nodes (above, s_l and $s_{x(i)}$) aligned to a pair of parent-child target-tree nodes (above, t_j and t_i). In deriving $\tau^{\mathbf{t},j}$, the model first chooses the configuration that will hold among t_i, t_j, $s_{x(i)}$ (which has yet to be chosen), and s_l (line 7). This is defined for configuration c log-linearly by:[5]

$$p_{config}(\mathsf{c} \mid t_j, l, \tau^{\mathbf{s}}) = \frac{\alpha_{\mathsf{c}}}{\sum_{\mathsf{c}': \exists s_k, config(t_i, t_j, s_k, s_l) = \mathsf{c}'} \alpha_{\mathsf{c}'}} \quad (14)$$

Permissible configurations in our model are shown in Table 1. These are identical to prior work (Smith and Eisner, 2006; Wang et al., 2007), except that we add a "root" configuration that aligns the target parent-child pair to null and the head word of the source sentence, respectively. Using many permissible configurations helps remove negative effects from noisy parses, which our learner treats as evidence. Fig. 1 shows some examples of major configurations that G_{p} discovers in the data.

Source tree alignment After choosing the configuration, the specific node in $\tau^{\mathbf{s}}$ that t_i will align to, $s_{x(i)}$ is drawn uniformly (line 8) from among those in the configuration selected.

Dependency label, POS, and named entity class The newly generated target word's dependency label, POS, and named entity class drawn from multinomial distributions p_{lab}, p_{pos}, and p_{ne} that condition, respectively, on the configuration and the POS and named entity class of the aligned source-tree word $s_{x(i)}$ (lines 9–11).

[5] We use log-linear models three times: for the configuration, the lexical semantics class, and the word. Each time, we are essentially assigning one weight per outcome and renormalizing among the subset of outcomes that are possible given what has been derived so far.

Configuration	Description
parent-child	$\tau_p^{\mathbf{s}}(x(i)) = x(j)$, appended with $\tau_\ell^{\mathbf{s}}(x(i))$
child-parent	$x(i) = \tau_p^{\mathbf{s}}(x(j))$, appended with $\tau_\ell^{\mathbf{s}}(x(j))$
grandparent-grandchild	$\tau_p^{\mathbf{s}}(\tau_p^{\mathbf{s}}(x(i))) = x(j)$, appended with $\tau_\ell^{\mathbf{s}}(x(i))$
siblings	$\tau_p^{\mathbf{s}}(x(i)) = \tau_p^{\mathbf{s}}(x(j)), x(i) \neq x(j)$
same-node	$x(i) = x(j)$
c-command	the parent of one source-side word is an ancestor of the other source-side word
root	$x(j) = 0, x(i)$ is the root of $\mathbf{s}$
child-null	$x(i) = 0$
parent-null	$x(j) = 0$, $x(i)$ is something other than root of $\mathbf{s}$
other	catch-all for all other types of configurations, which are permitted

Table 1: Permissible configurations. i is an index in $\mathbf{t}$ whose configuration is to be chosen; $j = \tau_p^{\mathbf{t}}(i)$ is i's parent.

WordNet relation(s) The model next chooses a lexical semantics relation between $s_{x(i)}$ and the yet-to-be-chosen word t_i (line 12). Following Wang et al. (2007),[6] we employ a 14-feature log-linear model over all logically possible combinations of the 14 WordNet relations (Miller, 1995).[7] Similarly to Eq. 14, we normalize this log-linear model based on the set of relations that are non-empty in WordNet for the word $s_{x(i)}$.

Word Finally, the target word is randomly chosen from among the set of words that bear the lexical semantic relationship just chosen (line 13). This distribution is, again, defined log-linearly:

$$p_{word}(t_i \mid lsrel(t_i) = \mathsf{R}, s_{x(i)}) = \frac{\alpha_{t_i}}{\sum_{w': s_{x(i)} \mathsf{R} w'} \alpha_{w'}} \quad (15)$$

Here α_w is the Good-Turing unigram probability estimate of a word w from the Gigaword corpus (Graff, 2003).

3.5 Base Grammar G_0

In addition to the QG that generates a second sentence bearing the desired relationship (paraphrase or not) to the first sentence $\mathbf{s}$, our model in §2 also requires a base grammar G_0 over $\mathbf{s}$.

We view this grammar as a trivial special case of the same QG model already described. G_0 assumes the empty source sentence consists only of

[6] Note that Wang et al. (2007) designed p_{kid} as an interpolation between a log-linear lexical semantics model and a word model. Our approach is more fully generative.

[7] These are: identical-word, synonym, antonym (including extended and indirect antonym), hypernym, hyponym, derived form, morphological variation (e.g., plural form), verb group, entailment, entailed-by, see-also, causal relation, whether the two words are same and is a number, and no relation.

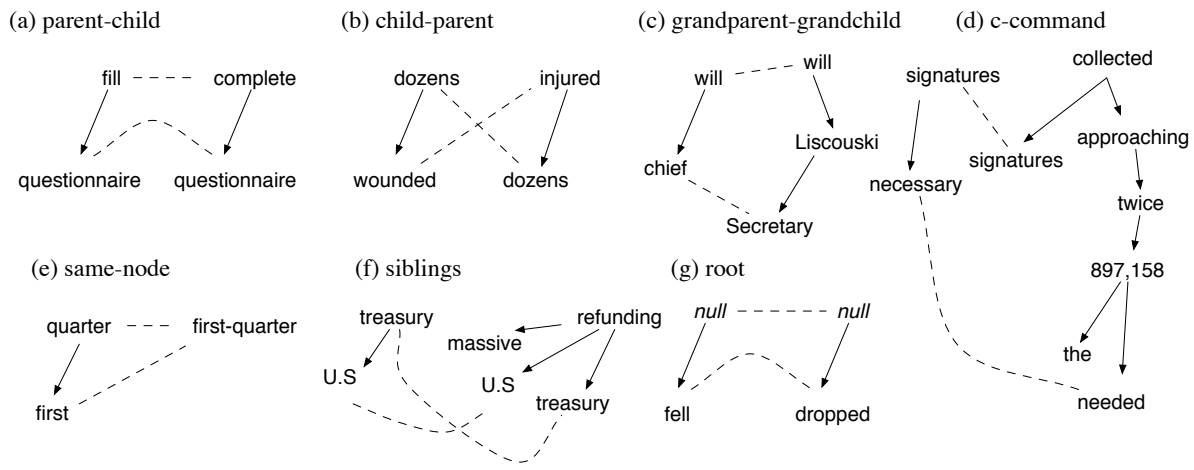

Figure 1: Some example configurations from Table 1 that G_{p} discovers in the dev. data. Directed arrows show head-modifier relationships, while dotted arrows show alignments.

a single wall node. Thus every word generated under G_0 aligns to null, and we can simplify the dynamic programming algorithm that scores a tree $\tau^{\mathbf{s}}$ under G_0:

$$\begin{aligned} C'(i) &= p_{val}(|\lambda^{\mathbf{t}}(i)|, |\rho^{\mathbf{t}}(i)| \mid s_i) \\ &\quad \times p_{lab}(\tau_\ell^{\mathbf{t}}(i)) \times p_{pos}(pos(t_i)) \times p_{ne}(ne(t_i)) \\ &\quad \times p_{word}(t_i) \times \textstyle\prod_{j:\tau^{\mathbf{t}}(j)=i} C'(j) \end{aligned} \tag{16}$$

where the final product is 1 when t_i has no children. It should be clear that $p(\mathbf{s} \mid G_0) = C'(0)$.

We estimate the distributions over dependency labels, POS tags, and named entity classes using the transformed treebank (footnote 4). The distribution over words is taken from the Gigaword corpus (as in §3.4).

It is important to note that G_0 is designed to give a smoothed estimate of the probability of a particular parsed, named entity-tagged sentence. It is never used for parsing or for generation; it is only used as a component in the generative probability model presented in §2 (Eq. 2).

3.6 Discriminative Training

Given training data $\left\langle \langle \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, c^{(i)} \rangle \right\rangle_{i=1}^{N}$, we train the model discriminatively by maximizing regularized conditional likelihood:

$$\max_{\Theta} \sum_{i=1}^{N} \log \underbrace{p_Q(c^{(i)} \mid \mathbf{s}_1^{(i)}, \mathbf{s}_2^{(i)}, \Theta)}_{\text{Eq. 2 relates this to } G_{\{0,\text{p},\text{n}\}}} - C\|\Theta\|_2^2 \tag{17}$$

The parameters Θ to be learned include the class priors, the conditional distributions of the dependency labels given the various configurations, the POS tags given POS tags, the NE tags given NE tags appearing in expressions 9–11, the configuration weights appearing in Eq. 14, and the weights of the various features in the log-linear model for the lexical-semantics model. As noted, the distributions p_{val}, the word unigram weights in Eq. 15, and the parameters of the base grammar are fixed using the treebank (see footnote 4) and the Gigaword corpus.

Since there is a hidden variable (x), the objective function is non-convex. We locally optimize using the L-BFGS quasi-Newton method (Liu and Nocedal, 1989). Because many of our parameters are multinomial probabilities that are constrained to sum to one and L-BFGS is not designed to handle constraints, we treat these parameters as unnormalized weights that get renormalized (using a softmax function) before calculating the objective.

4 Data and Task

In all our experiments, we have used the Microsoft Research Paraphrase Corpus (Dolan et al., 2004; Quirk et al., 2004). The corpus contains 5,801 pairs of sentences that have been marked as "equivalent" or "not equivalent." It was constructed from thousands of news sources on the web. Dolan and Brockett (2005) remark that this corpus was created semi-automatically by first training an SVM classifier on a disjoint annotated 10,000 sentence pair dataset and then applying the SVM on an unseen 49,375 sentence pair corpus, with its output probabilities skewed towards over-identification, i.e., towards generating some false paraphrases. 5,801 out of these 49,375 pairs were randomly selected and presented to human judges for refinement into true and false paraphrases. 3,900 of the pairs were marked as having

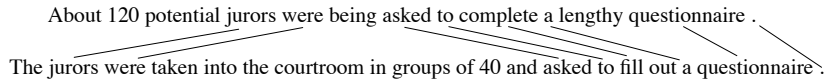

Figure 2: Discovered alignment of Ex. 19 produced by G_{p}. Observe that the model aligns identical words and also "complete" and "fill" in this specific case. This kind of alignment provides an edge over a simple lexical overlap model.

"mostly bidirectional entailment," a standard definition of the paraphrase relation. Each sentence was labeled first by two judges, who averaged 83% agreement, and a third judge resolved conflicts.

We use the standard data split into 4,076 (2,753 paraphrase, 1,323 not) training and 1,725 (1147 paraphrase, 578 not) test pairs. We reserved a randomly selected 1,075 training pairs for tuning.We cite some examples from the training set here:

(18) Revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

With the scandal hanging over Stewart's company, revenue in the first quarter of the year dropped 15 percent from the same period a year earlier.

(19) About 120 potential jurors were being asked to complete a lengthy questionnaire.

The jurors were taken into the courtroom in groups of 40 and asked to fill out a questionnaire.

Ex. 18 is a true paraphrase pair. Notice the high lexical overlap between the two sentences (unigram overlap of 100% in one direction and 72% in the other). Ex. 19 is another true paraphrase pair with much lower lexical overlap (unigram overlap of 50% in one direction and 30% in the other). Notice the use of similar-meaning phrases and irrelevant modifiers that retain the same meaning in both sentences, which a lexical overlap model cannot capture easily, but a model like a QG might. Also, in both pairs, the relationship cannot be called total bidirectional equivalence because there is some extra information in one sentence which cannot be inferred from the other.

Ex. 20 was labeled "not paraphrase":

(20) "There were a number of bureaucratic and administrative missed signals - there's not one person who's responsible here," Gehman said.

In turning down the NIMA offer, Gehman said, "there were a number of bureaucratic and administrative missed signals here.

There is significant content overlap, making a decision difficult for a naïve lexical overlap classifier. (In fact, p_Q labels this example n while the lexical overlap models label it p.)

The fact that negative examples in this corpus were selected because of their high lexical overlap is important. It means that any discriminative model is expected to learn to distinguish mere overlap from paraphrase. This seems appropriate, but it does mean that the "not paraphrase" relation ought to be denoted "not paraphrase but deceptively similar on the surface." It is for this reason that we use a special QG for the n relation.

5 Experimental Evaluation

Here we present our experimental evaluation using p_Q. We trained on the training set (3,001 pairs) and tuned model metaparameters (C in Eq. 17) and the effect of different feature sets on the development set (1,075 pairs). We report accuracy on the official MSRPC test dataset. If the posterior probability $p_Q(\mathsf{p} \mid \mathbf{s}_1, \mathbf{s}_2)$ is greater than 0.5, the pair is labeled "paraphrase" (as in Eq. 1).

5.1 Baseline

We replicated a state-of-the-art baseline model for comparison. Wan et al. (2006) report the best published accuracy, to our knowledge, on this task, using a support vector machine. Our baseline is a reimplementation of Wan et al. (2006), using features calculated directly from $\mathbf{s}_1$ and $\mathbf{s}_2$ without recourse to any hidden structure: proportion of word unigram matches, proportion of lemmatized unigram matches, BLEU score (Papineni et al., 2001), BLEU score on lemmatized tokens, F measure (Turian et al., 2003), difference of sentence length, and proportion of dependency relation overlap. The SVM was trained to classify positive and negative examples of paraphrase using SVM^{light} (Joachims, 1999).[8] Metaparameters, tuned on the development data, were the regularization constant and the degree of the polynomial kernel (chosen in $[10^{-5}, 10^{2}]$ and 1–5 respectively.).[9]

It is unsurprising that the SVM performs very well on the MSRPC because of the corpus creation process (see Sec. 4) where an SVM was applied as well, with very similar features and a skewed decision process (Dolan and Brockett, 2005).

[8] `http://svmlight.joachims.org`

[9] Our replication of the Wan et al. model is approximate, because we used different preprocessing tools: MXPOST for POS tagging (Ratnaparkhi, 1996), MSTParser for parsing (McDonald et al., 2005), and Dan Bikel's interface (`http://www.cis.upenn.edu/~dbikel/software.html#wn`) to WordNet (Miller, 1995) for lemmatization information. Tuning led to $C = 17$ and polynomial degree 4.

	Model	Accuracy	Precision	Recall
baselines	all p	66.49	66.49	100.00
	Wan et al. SVM (reported)	75.63	77.00	90.00
	Wan et al. SVM (replication)	75.42	76.88	90.14
p_Q	lexical semantics features removed	68.64	68.84	96.51
	all features	73.33	74.48	91.10
	c-command disallowed (best; see text)	73.86	74.89	91.28
§6	p_L	75.36	78.12	87.44
	product of experts	76.06	79.57	86.05
oracles	Wan et al. SVM and p_L	*80.17*	*100.00*	*92.07*
	Wan et al. SVM and p_Q	*83.42*	*100.00*	*96.60*
	p_Q and p_L	*83.19*	*100.00*	*95.29*

Table 2: Accuracy, p-class precision, and p-class recall on the test set (N = 1,725). See text for differences in implementation between Wan et al. and our replication; their reported score does not include the full test set.

5.2 Results

Tab. 2 shows performance achieved by the baseline SVM and variations on p_Q on the test set. We performed a few feature ablation studies, evaluating on the development data. We removed the lexical semantics component of the QG,[10] and disallowed the syntactic configurations one by one, to investigate which components of p_Q contributes to system performance. The lexical semantics component is critical, as seen by the drop in accuracy from the table (without this component, p_Q behaves almost like the "all p" baseline). We found that the most important configurations are "parent-child," and "child-parent" while damage from ablating other configurations is relatively small. Most interestingly, disallowing the "c-command" configuration resulted in the best absolute accuracy, giving us the best version of p_Q. The c-command configuration allows more distant nodes in a source sentence to align to parent-child pairs in a target (see Fig. 1d). Allowing this configuration guides the model in the wrong direction, thus reducing test accuracy. We tried disallowing more than one configuration at a time, without getting improvements on development data. We also tried ablating the WordNet relations, and observed that the "identical-word" feature hurt the model the most. Ablating the rest of the features did not produce considerable changes in accuracy.

The development data-selected p_Q achieves higher recall by 1 point than Wan et al.'s SVM, but has precision 2 points worse.

5.3 Discussion

It is quite promising that a linguistically-motivated probabilistic model comes so close to a string-similarity baseline, *without* incorporating string-local phrases. We see several reasons to prefer the more intricate QG to the straightforward SVM. First, the QG discovers hidden alignments between words. Alignments have been leveraged in related tasks such as textual entailment (Giampiccolo et al., 2007); they make the model more interpretable in analyzing system output (e.g., Fig. 2). Second, the paraphrases of a sentence can be considered to be monolingual translations. We model the paraphrase problem using a direct machine translation model, thus providing a translation interpretation of the problem. This framework could be extended to permit paraphrase *generation*, or to exploit other linguistic annotations, such as representations of semantics (see, e.g., Qiu et al., 2006).

Nonetheless, the usefulness of surface overlap features is difficult to ignore. We next provide an efficient way to combine a surface model with p_Q.

6 Product of Experts

Incorporating structural alignment and surface overlap features inside a single model can make exact inference infeasible. As an example, consider features like n-gram overlap percentages that provide cues of content overlap between two sentences. One intuitive way of including these features in a QG could be including these only at the root of the target tree, i.e. while calculating $C(r, 0)$. These features have to be included in estimating p_{kid}, which has log-linear component models (Eq. 7- 13). For these bigram or trigram overlap features, a similar log-linear model has to be normalized with a partition function, which considers the (unnormalized) scores of *all* possible target sentences, given the source sentence.

We therefore combine p_Q with a lexical overlap model that gives another posterior probability estimate $p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)$ through a product of experts (PoE; Hinton, 2002), $p_J(c \mid \mathbf{s}_1, \mathbf{s}_2)$

$$= \frac{p_Q(c \mid \mathbf{s}_1, \mathbf{s}_2) \times p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)}{\sum_{c' \in \{\mathsf{p},\mathsf{n}\}} p_Q(c' \mid \mathbf{s}_1, \mathbf{s}_2) \times p_L(c' \mid \mathbf{s}_1, \mathbf{s}_2)} \quad (21)$$

[10] This is accomplished by eliminating lines 12 and 13 from the definition of p_{kid} and redefining p_{word} to be the unigram word distribution estimated from the Gigaword corpus, as in G_0, without the help of WordNet.

Eq. 21 takes the product of the two models' posterior probabilities, then normalizes it to sum to one. PoE models are used to efficiently combine several expert models that individually constrain different dimensions in high-dimensional data, the product therefore constraining all of the dimensions. Combining models in this way grants to each expert component model the ability to "veto" a class by giving it low probability; the most probable class is the one that is least objectionable to all experts.

Probabilistic Lexical Overlap Model We devised a logistic regression (LR) model incorporating 18 simple features, computed directly from $\mathbf{s}_1$ and $\mathbf{s}_2$, without modeling any hidden correspondence. LR (like the QG) provides a probability distribution, but uses surface features (like the SVM). The features are of the form $precision_n$ (number of n-gram matches divided by the number of n-grams in $\mathbf{s}_1$), $recall_n$ (number of n-gram matches divided by the number of n-grams in $\mathbf{s}_2$) and F_n (harmonic mean of the previous two features), where $1 \leq n \leq 3$. We also used lemmatized versions of these features. This model gives the posterior probability $p_L(c \mid \mathbf{s}_1, \mathbf{s}_2)$, where $c \in \{\mathsf{p}, \mathsf{n}\}$. We estimated the model parameters analogously to Eq. 17. Performance is reported in Tab. 2; this model is on par with the SVM, though trading recall in favor of precision. We view it as a probabilistic simulation of the SVM more suitable for combination with the QG.

Training the PoE Various ways of training a PoE exist. We first trained p_Q and p_L separately as described, then initialized the PoE with those parameters. We then continued training, maximizing (unregularized) conditional likelihood.

Experiment We used p_Q with the "c-command" configuration excluded, and the LR model in the product of experts. Tab. 2 includes the final results achieved by the PoE. The PoE model outperforms all the other models, achieving an accuracy of 76.06%.[11] The PoE is conservative, labeling a pair as p only if the LR and the QG give it strong p probabilities. This leads to high precision, at the expense of recall.

Oracle Ensembles Tab. 2 shows the results of three different oracle ensemble systems that correctly classify a pair if *either* of the two individual systems in the combination is correct. Note that the combinations involving p_Q achieve 83%, the human agreement level for the MSRPC. The LR and SVM are highly similar, and their oracle combination does not perform as well.

[11] This accuracy is significant over p_Q under a paired t-test ($p < 0.04$), but is not significant over the SVM.

7 Related Work

There is a growing body of research that uses the MSRPC (Dolan et al., 2004; Quirk et al., 2004) to build models of paraphrase. As noted, the most successful work has used edit distance (Zhang and Patrick, 2005) or bag-of-words features to measure sentence similarity, along with shallow syntactic features (Finch et al., 2005; Wan et al., 2006; Corley and Mihalcea, 2005). Qiu et al. (2006) used predicate-argument annotations.

Most related to our approach, Wu (2005) used inversion transduction grammars—a synchronous context-free formalism (Wu, 1997)—for this task. Wu reported only positive-class (p) precision (not accuracy) on the test set. He obtained 76.1%, while our PoE model achieves 79.6% on that measure. Wu's model can be understood as a strict hierarchical maximum-alignment method. In contrast, our alignments are soft (we sum over them), and we do not require strictly isomorphic syntactic structures. Most importantly, our approach is founded on a stochastic generating process and estimated discriminatively for this task, while Wu did not estimate any parameters from data at all.

8 Conclusion

In this paper, we have presented a probabilistic model of paraphrase incorporating syntax, lexical semantics, and hidden loose alignments between two sentences' trees. Though it fully defines a generative process for both sentences and their relationship, the model is discriminatively trained to maximize conditional likelihood. We have shown that this model is competitive for determining whether there exists a semantic relationship between them, and can be improved by principled combination with more standard lexical overlap approaches.

Acknowledgments

The authors thank the three anonymous reviewers for helpful comments and Alan Black, Frederick Crabbe, Jason Eisner, Kevin Gimpel, Rebecca Hwa, David Smith, and Mengqiu Wang for helpful discussions. This work was supported by DARPA grant NBCH-1080004.

References

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proc. of NAACL*.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved statistical machine translation using paraphrases. In *Proc. of HLT-NAACL*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proc. of ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources. In *Proc. of COLING*.

Andrew Finch, Young Sook Hwang, and Eiichiro Sumita. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *Proc. of IWP*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

David Graff. 2003. English Gigaword. Linguistic Data Consortium.

Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming (Ser. B)*, 45(3):503–528.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proc. of EWNLG*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Kathleen R. McKeown. 1979. Paraphrasing using given and new information in a question-answer system. In *Proc. of ACL*.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *Proc. of ACL*.

George A. Miller. 1995. Wordnet: a lexical database for English. *Commun. ACM*, 38(11):39–41.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proc. of EMNLP*.

Chris Quirk, Chris Brockett, and William B. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proc. of EMNLP*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*.

David A. Smith and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*.

Joseph P. Turian, Luke Shen, and I. Dan Melamed. 2003. Evaluation of machine translation and its evaluation. In *Proc. of Machine Translation Summit IX*.

Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. 2006. Using dependency-based features to take the "para-farce" out of paraphrase. In *Proc. of ALTW*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Comput. Linguist.*, 23(3).

Dekai Wu. 2005. Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *Proc. of ALTW*.