Combined Distributional and Logical Semantics

Mike Lewis
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
mike.lewis@ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

We introduce a new approach to semantics which combines the benefits of distributional and formal logical semantics. Distributional models have been successful in modelling the meanings of content words, but logical semantics is necessary to adequately represent many function words. We follow formal semantics in mapping language to logical representations, but differ in that the relational constants used are induced by offline distributional clustering at the level of predicate-argument structure. Our clustering algorithm is highly scalable, allowing us to run on corpora the size of Gigaword. Different senses of a word are disambiguated based on their induced types. We outperform a variety of existing approaches on a wide-coverage question answering task, and demonstrate the ability to make complex multi-sentence inferences involving quantifiers on the FraCaS suite.

1 Introduction

Mapping natural language to meaning representations is a central challenge of NLP. There has been much recent progress in unsupervised distributional semantics, in which the meaning of a word is induced based on its usage in large corpora. This approach is useful for a range of key applications including question answering and relation extraction (Lin and Pantel, 2001; Poon and Domingos, 2009; Yao et al., 2011). Because such a semantics can be automically induced, it escapes the limitation of depending on relations from hand-built training data, knowledge bases or ontologies, which have proved of limited use in capturing the huge variety of meanings that can be expressed in language.

However, distributional semantics has largely developed in isolation from the formal semantics literature. Whilst distributional semantics has been effective in modelling the meanings of content words such as nouns and verbs, it is less clear that it can be applied to the meanings of function words. Semantic operators, such as determiners, negation, conjunctions, modals, tense, mood, aspect, and plurals are ubiquitous in natural language, and are crucial for high performance on many practical applications—but current distributional models struggle to capture even simple examples. Conversely, computational models of formal semantics have shown low recall on practical applications, stemming from their reliance on ontologies such as WordNet (Miller, 1995) to model the meanings of content words (Bobrow et al., 2007; Bos and Markert, 2005).

For example, consider what is needed to answer a question like *Did Google buy YouTube?* from the following sentences:

1. Google purchased YouTube
2. Google's acquisition of YouTube
3. Google acquired every company
4. YouTube may be sold to Google
5. Google will buy YouTube or Microsoft
6. Google didn't takeover YouTube

All of these require knowledge of lexical semantics (e.g. that *buy* and *purchase* are synonyms), but some also need interpretation of quantifiers, negatives, modals and disjunction. It seems unlikely that

Transactions of the Association for Computational Linguistics, 1 (2013) 179–192. Action Editor: Johan Bos.
Submitted 1/2013; Revised 3/2013; Published 5/2013. ©2013 Association for Computational Linguistics.

distributional or formal approaches can accomplish the task alone.

We propose a method for mapping natural language to first-order logic representations capable of capturing the meanings of function words such as *every*, *not* and *or*, but which also uses distributional statistics to model the meaning of content words. Our approach differs from standard formal semantics in that the non-logical symbols used in the logical form are cluster identifiers. Where standard semantic formalisms would map the verb *write* to a *write'* symbol, we map it to a cluster identifier such as *relation37*, which the noun *author* may also map to. This mapping is learnt by offline clustering.

Unlike previous distributional approaches, we perform clustering at the level of predicate-argument structure, rather than syntactic dependency structure. This means that we abstract away from many syntactic differences that are not present in the semantics, such as conjunctions, passives, relative clauses, and long-range dependencies. This significantly reduces sparsity, so we have fewer predicates to cluster and more observations for each.

Of course, many practical inferences rely heavily on background knowledge about the world—such knowledge falls outside the scope of this work.

2 Background

Our approach is based on Combinatory Categorial Grammar (CCG; Steedman, 2000), a strongly lexicalised theory of language in which lexical entries for words contain all language-specific information. The lexical entry for each word contains a *syntactic category*, which determines which other categories the word may combine with, and a *semantic interpretation*, which defines the compositional semantics. For example, the lexicon may contain the entry:
$write \vdash (S\backslash NP)/NP : \lambda y \lambda x.write'(x,y)$

Crucially, there is a transparent interface between the syntactic category and the semantics. For example the transitive verb entry above defines the verb syntactically as a function mapping two noun-phrases to a sentence, and semantically as a binary relation between its two argument entities. This means that it is relatively straightforward to deterministically map parser output to a logical form, as in the Boxer system (Bos, 2008). This form of semantics captures the underlying predicate-argument structure, but fails to license many important inferences—as, for example, $write$ and $author$ do not map to the same predicate.

In addition to the lexicon, there is a small set of binary combinators and unary rules, which have a syntactic and semantic interpretation. Figure 1 gives an example CCG derivation.

$$\begin{array}{ccc}
\text{Every} & \text{dog} & \text{barks} \\
\hline
NP^{\uparrow}/N & N & S\backslash NP \\
\lambda p \lambda q.\forall x[p(x) \implies q(x)] & \lambda x.dog'(x) & \lambda x.bark'(x) \\
\hline
\multicolumn{2}{c}{NP^{\uparrow} \quad >} & \\
\multicolumn{2}{c}{\lambda q.\forall x[dog'(x) \implies q(x)]} & \\
\hline
\multicolumn{3}{c}{S \quad >} \\
\multicolumn{3}{c}{\forall x[dog'(x) \implies bark'(x)]}
\end{array}$$

Figure 1: A standard logical form derivation using CCG. The $NP^{\uparrow}$ notation means that the subject is type-raised, and taking the verb-phrase as an argument—so is an abbreviation of $S/(S\backslash NP)$. This is necessary in part to support a correct semantics for quantifiers.

Input Sentence
Shakespeare wrote Macbeth
⇓
Intial semantic analysis
$write_{arg0,arg1}$*(shakespeare, macbeth)*
⇓
Entity Typing
$write_{arg0:PER,arg1:BOOK}$*(shakespeare:PER, macbeth:BOOK)*
⇓
Distributional semantic analysis
relation37(shakespeare:PER, macbeth:BOOK)

Figure 2: Layers used in our model.

3 Overview of Approach

We attempt to learn a CCG lexicon which maps equivalent words onto the same logical form—for example learning entries such as:
$author \vdash N/PP[of] : \lambda x \lambda y.relation37(x,y)$
$write \vdash (S\backslash NP)/NP : \lambda x \lambda y.relation37(x,y)$
The only change to the standard CCG derivation is that the symbols used in the logical form are arbitrary relation identifiers. We learn these by first mapping to a deterministic logical form (using predicates

such as *author'* and *write'*), using a process similar to Boxer, and then clustering predicates based on their arguments. This lexicon can then be used to parse new sentences, and integrates seamlessly with CCG theories of formal semantics.

Typing predicates—for example, determining that *writing* is a relation between people and books—has become standard in relation clustering (Schoenmackers et al., 2010; Berant et al., 2011; Yao et al., 2012). We demonstate how to build a typing model into the CCG derivation, by subcategorizing all terms representing entities in the logical form with a more detailed type. These types are also induced from text, as explained in Section 5, but for convenience we describe them with human-readable labels, such as *PER*, *LOC* and *BOOK*.

A key advantage of typing is that it allows us to model ambiguous predicates. Following Berant et al. (2011), we assume that different type signatures of the same predicate have different meanings, but given a type signature a predicate is unambiguous. For example a different lexical entry for the verb *born* is used in the contexts *Obama was born in Hawaii* and *Obama was born in 1961*, reflecting a distinction in the semantics that is not obvious in the syntax[1]. Typing also greatly improves the efficiency of clustering, as we only need to compare predicates with the same type during clustering (for example, we do not have to consider clustering a predicate between people and places with predicates between people and dates).

In this work, we focus on inducing binary relations. Many existing approaches have shown how to produce good clusterings of (non-event) nouns (Brown et al., 1992), any of which could be simply integrated into our semantics—but relation clustering remains an open problem (see Section 9). N-ary relations are binarized, by creating a binary relation between each pair of arguments. For example, for the sentence *Russia sold Alaska to the United States*, the system creates three binary relations— corresponding to *sellToSomeone(Russia, Alaska)*, *buyFromSomeone(US, Alaska)*, *sellSomethingTo(Russia, US)*. This transformation does not exactly preserve meaning, but still captures the most important relations. Note that this allows us to compare semantic relations across different syntactic types—for example, both transitive verbs and argument-taking nouns can be seen as expressing binary semantic relations between entities.

Figure 2 shows the layers used in our model.

4 Initial Semantic Analysis

The initial semantic analysis maps parser output onto a logical form, in a similar way to Boxer. The semantic formalism is based on Steedman (2012).

The first step is syntactic parsing. We use the C&C parser (Clark and Curran, 2004), trained on CCGBank (Hockenmaier and Steedman, 2007), using the refined version of Honnibal et al. (2010) which brings the syntax closer to the predicate-argument structure. An automatic post-processing step makes a number of minor changes to the parser output, which converts the grammar into one more suitable for our semantics. PP (prepositional phrase) and PR (phrasal verb complement) categories are sub-categorised with the relevant preposition. Noun compounds with the same MUC named-entity type (Chinchor and Robinson, 1997) are merged into a single non-compositional node[2] (we otherwise ignore named-entity types). All argument NPs and PPs are type-raised, allowing us to represent quantifiers. All prepositional phrases are treated as core arguments (i.e. given the category *PP*, not adjunct categories like $(N\backslash N)/NP$ or $((S\backslash NP)\backslash(S\backslash NP))/NP$), as it is difficult for the parser to distinguish arguments and adjuncts.

Initial semantic lexical entries for almost all words can be generated automatically from the syntactic category and POS tag (obtained from the parser), as the syntactic category captures the underlying predicate-argument structure. We use a Davidsonian-style representation of arguments (Davidson, 1967), which we binarize by creating a separate predicate for each pair of arguments of a word. These predicates are labelled with the lemma of the head word and a Propbank-style argument key (Kingsbury and Palmer, 2002), e.g. *arg0*, *argIn*. We distinguish noun and verb predicates based on POS

[1]Whilst this assumption is very useful, it does not always hold—for example, the genitive in *Shakespeare's book* is ambiguous between ownership and authorship relations even given the types of the arguments.

[2]For example, this allows us to give *Barack Obama* the semantics $\lambda x.barack\_obama(x)$ instead of $\lambda x.barack(x) \wedge obama(x)$, which is more convenient for collecting distributional statistics.

	Word	Category	Semantics
Automatic	author	$N/PP[of]$	$\lambda x \lambda y.author_{arg0,argOf}(y,x)$
	write	$(S\backslash NP)/NP$	$\lambda x \lambda y.write_{arg0,arg1}(y,x)$
Manual	every	$NP^{\uparrow}/N$	$\lambda p \lambda q.\forall x[p(x) \rightarrow q(x)]$
	not	$(S\backslash NP)/(S\backslash NP)$	$\lambda p \lambda x.\neg p(x)$

Figure 3: Example initial lexical entries

tag—so, for example, we have different predicates for *effect* as a noun or verb.

This algorithm can be overridden with manual lexical entries for specific closed-class function words. Whilst it may be possible to learn these from data, our approach is pragmatic as there are relatively few such words, and the complex logical forms required would be difficult to induce from distributional statistics. We add a small number of lexical entries for words such as negatives (*no*, *not* etc.), and quantifiers (numbers, *each*, *every*, *all*, etc.).

Some example initial lexical entries are shown in Figure 3.

5 Entity Typing Model

Our entity-typing model assigns types to nouns, which is useful for disambiguating polysemous predicates. Our approach is similar to O'Seaghdha (2010) in that we aim to cluster entities based on the noun and unary predicates applied to them (it is simple to convert from the binary predicates to unary predicates). For example, we want the pair *(born$_{argIn}$, 1961)* to map to a DAT type, and *(born$_{argIn}$, Hawaii)* to map to a LOC type. This is non-trivial, as both the predicates and arguments can be ambiguous between multiple types—but topic models offer a good solution (described below).

5.1 Topic Model

We assume that the type of each argument of a predicate depends only on the predicate and argument, although Ritter et al. (2010) demonstrate an advantage of modelling the joint probability of the types of multiple arguments of the same predicate. We use the standard Latent Dirichlet Allocation model (Blei et al., 2003), which performs comparably to more complex models proposed in O'Seaghdha (2010).

In topic-modelling terminology, we construct a document for each unary predicate (e.g. *born$_{argIn}$*), based on all of its argument entities (words). We assume that these arguments are drawn from a small number of types (topics), such as PER, DAT or LOC[3]. Each type j has a multinomial distribution ϕ_j over arguments (for example, a LOC type is more likely to generate *Hawaii* than *1961*). Each unary predicate i has a multinomial distribution θ_i over topics, so the *born$_{argIn}$* predicate will normally generate a DAT or LOC type. Sparse Dirichlet priors α and β on the multinomials bias the distributions to be peaky. The parameters are estimated by Gibbs sampling, using the Mallet implementation (McCallum, 2002).

The generative story to create the data is:

```
For every type $k$:
  Draw the $p(arg|k)$ distribution $\phi_k$ from $Dir(\beta)$
For every unary predicate $i$:
  Draw the $p(type|i)$ distribution $\theta_i$ from $Dir(\alpha)$
  For every argument $j$:
    Draw a type $z_{ij}$ from $Mult(\theta_i)$
    Draw an argument $w_{ij}$ from $Mult(\phi_{\theta_i})$
```

5.2 Typing in Logical Form

In the logical form, all constants and variables representing entities x can be assigned a distribution over types $p_x(t)$ using the type model. An initial type distribution is applied in the lexicon, using the ϕ distributions for the types of nouns, and the θ_i distributions for the type of arguments of binary predicates (inverted using Bayes' rule). Then at each β-reduction in the derivation, we update probabilities of the types to be the product of the type distributions of the terms being reduced. If two terms x and

[3]Types are induced from the text, but we give human-readable labels here for convenience.

$$\begin{array}{cc}
\text{file} & \text{a suit} \\
\hline
(S\backslash NP)/NP & NP^{\uparrow} \\
\lambda y\!:\!\left\{\begin{array}{c} DOC\!=\!0.5 \\ LEGAL\!=\!0.4 \\ CLOTHES\!=\!0.01 \\ \ldots \end{array}\right\}\lambda x\!:\!\left\{\begin{array}{c} PER=0.7 \\ ORG=0.2 \\ \ldots \end{array}\right\}.file_{arg0,arg1}(x,y) & \lambda p.\exists y\!:\!\left\{\begin{array}{c} CLOTHES=0.6 \\ LEGAL=0.3 \\ DOC\!=\!0.001 \\ \ldots \end{array}\right\}[suit'(y)\wedge p(y)] \\
\hline
\multicolumn{2}{c}{S\backslash NP \qquad <} \\
\multicolumn{2}{c}{\lambda x\!:\left\{\begin{array}{c} PER=0.7 \\ ORG=0.2 \\ \ldots \end{array}\right\}\exists y\!:\left\{\begin{array}{c} LEGAL=0.94 \\ CLOTHES=0.05 \\ DOC=0.004 \\ \ldots \end{array}\right\})[suit'(y)\wedge file_{arg0,arg1}(x,y)]}
\end{array}$$

Figure 4: Using the type model for disambiguation in the derivation of *file a suit*. Type distributions are shown after the variable declarations. Both *suit* and the object of *file* are lexically ambiguous between different types, but after the β-reduction only one interpretation is likely. If the verb were *wear*, a different interpretation would be preferred.

y combine to a term z:

$$p_z(t) = \frac{p_x(t)p_y(t)}{\sum_{t'} p_x(t')p_y(t')}$$

For example, in *wore a suit* and *file a suit*, the variable representing *suit* may be lexically ambiguous between CLOTHES and LEGAL types, but the variables representing the objects of *wear* and $file$ will have preferences that allow us to choose the correct type when the terms combine. Figure 4 shows an example derivation using the type model for disambiguation[4].

6 Distributional Relation Clustering Model

The typed binary predicates can be grouped into clusters, each of which represents a distinct semantic relation. Note that because we cluster typed predicates, $born_{arg0:PER,argIn:LOC}$ and $born_{arg0:PER,argIn:DAT}$ can be clustered separately.

6.1 Corpus statistics

Typed binary predicates are clustered based on the expected number of times they hold between each argument-pair in the corpus. This means we create a single vector of argument-pair counts for each predicate (not a separate vector for each argument). For example, the vector for the typed predicate $write_{arg0:PER,arg1:BOOK}$ may contain non-zero counts for entity-pairs such as *(Shakespeare, Macbeth)*, *(Dickens, Oliver Twist)* and *(Rowling, Harry Potter)*. The entity-pair counts for $author_{arg0:PER,argOf:BOOK}$ may be similar, on the assumption that both are samples from the same underlying semantic relation.

[4]Our implementation follows Steedman (2012) in using Generalized Skolem Terms rather than existential quantifiers, in order to capture quantifier scope alternations monotonically, but we omit these from the example to avoid introducing new notation.

To find the expected number of occurrences of argument-pairs for typed binary predicates in a corpus, we first apply the type model to the derivation of each sentence, as described in Section 5.2. This outputs untyped binary predicates, with distributions over the types of their arguments. The type of the predicate must match the type of its arguments, so the type distribution of a binary predicate is simply the joint distribution of the two argument type distributions.

For example, if the arguments in a $born_{arg0,argIn}(obama, hawaii)$ derivation have the respective type distributions *(PER=0.9, LOC=0.1)* and *(LOC=0.7, DAT=0.3)*, the distribution over binary typed predicates is ($born_{arg0:PER,argIn:LOC}$=0.63, $born_{arg0:PER,argIn:DAT}$=0.27, etc.) The expected counts for $(obama, hawaii)$ in the vectors for $born_{arg0:PER,argIn:LOC}$ and $born_{arg0:PER,argIn:DAT}$ are then incremented by these probabilities.

6.2 Clustering

Many algorithms have been proposed for clustering predicates based on their arguments (Poon and Domingos, 2009; Yao et al., 2012). The number of relations in the corpus is unbounded, so the clustering algorithm should be non-parametric. It is also important that it remains tractable for very large numbers of predicates and arguments, in order to give us a greater coverage of language than can be achieved by hand-built ontologies.

We cluster the typed predicate vectors using the Chinese Whispers algorithm (Biemann, 2006)—

although somewhat ad-hoc, it is both non-parametric and highly scalable[5]. This has previously been used for noun-clustering by Fountain and Lapata (2011), who argue it is a cognitively plausible model for language acquisition. The collection of predicates and arguments is converted into a graph with one node per predicate, and edge weights representing the similarity between predicates. Predicates with different types have zero-similarity, and otherwise similarity is computed as the cosine-similarity of the tf-idf vectors of argument-pairs. We prune nodes occurring fewer than 20 times, edges with weights less than 10^{-3}, and a short list of stop predicates.

The algorithm proceeds as follows:

1. Each predicate p is assigned to a different semantic relation r_p
2. Iterate over the predicates p in a random order
3. Set $r_p = \arg\max_r \sum_{p'} \mathbb{1}_{r=r_{p'}} sim(p,p')$, where $sim(p,p')$ is the distributional similarity between p and p', and $\mathbb{1}_{r=r'}$ is 1 iff r=r' and 0 otherwise.
4. Repeat (2.) to convergence.

7 Semantic Parsing using Relation Clusters

The final phase is to use our relation clusters in the lexical entries of the CCG semantic derivation. This is slightly complicated by the fact that our predicates are lexically ambiguous between all the possible types they could take, and hence the relations they could express. For example, the system cannot tell whether *born in* is expressing a *birthplace* or *birthdate* relation until later in the derivation, when it combines with its arguments. However, all the possible logical forms are identical except for the symbols used, which means we can produce a packed logical form capturing the full distribution over logical forms. To do this, we make the predicate a function from argument types to relations.

For each word, we first take the lexical semantic definition produced by the algorithm in Section 4. For binary predicates in this definition (which will be untyped), we perform a deterministic lookup in the cluster model learned in Section 6, using all possible corresponding typed predicates. This allows us to represent the binary predicates as *packed predicates*: functions from argument types to relations.

For example, if the clustering maps $born_{arg0:PER,argIn:LOC}$ to *rel49* ("birthplace") and $born_{arg0:PER,argIn:DAT}$ to *rel53* ("birthdate"), our lexicon contains the following packed lexical entry (type-distributions on the variables are suppressed):

$$born \vdash (S\backslash NP)/PP[in] : \\ \lambda y \lambda x. \begin{Bmatrix} (x:PER, y:LOC) \Rightarrow rel49 \\ (x:PER, y:DAT) \Rightarrow rel53 \end{Bmatrix} (x,y)$$

The distributions over argument types then imply a distribution over relations. For example, if the packed-predicate for $born_{arg0,argIn}$ is applied to arguments *Obama* and *Hawaii*, with respective type distributions *(PER=0.9, LOC=0.1)* and *(LOC=0.7, DAT=0.3)*[6], the distribution over relations will be (*rel49*=0.63, *rel53*=0.27, etc.).

If *1961* has a type-distribution *(LOC=0.1, DAT=0.9)*, the output packed-logical form for *Obama was born in Hawaii in 1961* will be:

$$\begin{Bmatrix} rel49{=}0.63 \\ rel53{=}0.27 \\ \ldots \end{Bmatrix} (ob, hw) \wedge \begin{Bmatrix} rel49{=}0.09 \\ rel53{=}0.81 \\ \ldots \end{Bmatrix} (ob, 1961)$$

The probability of a given logical form can be read from this packed logical form.

8 Experiments

Our approach aims to offer a strong model of both formal and lexical semantics. We perform two evaluations, aiming to target each of these separately, but using the same semantic representations in each.

We train our system on Gigaword (Graff et al., 2003), which contains around 4 billion words of newswire. The type-model is trained using 15 types[7], and 5,000 iterations of Gibbs sampling (using the distributions from the final sample). Table 1

[5] We also experimented with a Dirichlet Process Mixture Model (Neal, 2000), but even with the efficient A* search algorithms introduced by Daumé III (2007), the cost of inference was found to be prohibitively high when run at large scale.

[6] These distributions are composed from the type-distributions for both the predicate and argument, as explained in Section 5

[7] This number was chosen by examination of models trained with different numbers of types. The algorithm produces semantically coherent clusters for much larger numbers of types, but many of these are fine-grained categories of people, which introduces sparsity in the relation clustering.

Type	Top Words
1	suspect, assailant, fugitive, accomplice
2	author, singer, actress, actor, dad
5	city, area, country, region, town, capital
8	subsidiary, automaker, airline, Co., GM
10	musical, thriller, sequel, special

Table 1: Most probable terms in some clusters induced by the Type Model.

shows some example types. The relation clustering uses only proper nouns, to improve precision (sparsity problems are partly offset by the large input corpus). Aside from parsing, the pipeline takes around a day to run using 12 cores.

8.1 Question Answering Experiments

As yet, there is no standard way of evaluating lexical semantics. Existing tasks like Recognising Textual Entailment (RTE; Dagan et al., 2006) rely heavily on background knowledge, which is beyond the scope of this work. Intrinsic evaluations of entailment relations have low inter-annotator agreement (Szpektor et al., 2007), due to the difficulty of evaluating relations out of context.

Our evaluation is based on that performed by Poon and Domingos (2009). We automatically construct a set of questions by sampling from text, and then evaluate how many answers can be found in a different corpus. From dependency-parsed newswire, we sample either $X \overset{nsubj}{\leftarrow} verb \overset{dobj}{\rightarrow} Y$, $X \overset{nsubj}{\leftarrow} verb \overset{pobj}{\rightarrow} Y$ or $X \overset{nsubj}{\leftarrow} be \overset{dobj}{\rightarrow} noun \overset{pobj}{\rightarrow} Y$ patterns, where X and Y are proper nouns and the verb is not on a list of stop verbs, and deterministically convert these to questions. For example, from *Google bought YouTube* we create the questions *What did Google buy?* and *What bought YouTube?*. The task is to find proper-noun answers to these questions in a different corpus, which are then evaluated by human annotators based on the sentence the answer was retrieved from[8]. Systems can return multiple answers to the same question (e.g. *What did Google buy?* may have many valid answers), and all of these contribute to the result. As none of the systems model tense or temporal semantics, annotators were instructed to annotate answers as correct if they were true at any time. This approach means we evaluate on relations in proportion to corpus frequency. We sample 1000 questions from the New York Times subset of Gigaword from 2010, and search for answers in the New York Times from 2009.

We evaluate the following approaches:

- **CCG-Baseline** The logical form produced by our CCG derivation, without the clustering.
- **CCG-WordNet** The CCG logical form, plus WordNet as a model of lexical semantics.
- **CCG-Distributional** The logical form including the type model and clusters.
- **Relational LDA** An LDA based model for clustering dependency paths (Yao et al., 2011). We train on New York Times subset of Gigaword[9], using their setup of 50 iterations with 100 relation types.
- **Reverb** A sophisticated Open Information Extraction system (Fader et al., 2011).

Unsupervised Semantic Parsing (USP; Poon and Domingos, 2009; USP; Poon and Domingos, 2010; USP; Titov and Klementiev, 2011) would be another obvious baseline. However, memory requirements mean it is not possible to run at this scale (our system is trained on 4 orders of magnitude more data than the USP evaluation). Yao et al. (2011) found it had comparable performance to Relational LDA.

For the CCG models, rather than performing full first-order inference on a large corpus, we simply test whether the question predicate subsumes a candidate answer predicate, and whether the arguments match[10]. In the case of CCG-Distributional, we calculate the probability that the two packed-predicates

[8]Common nouns are filtered automatically. To focus on evaluating the semantics, annotators ignored garbled sentences due to errors pre-processing the corpus (these are excluded from the results). We also automatically exclude weekday and month answers, which are overwhelmingly syntax errors for all systems—e.g. treating *Tuesday* as an object in *Obama announced Tuesday that...*

[9]This is around 35% of Gigaword, and was the largest scale possible on our resources.

[10]We do this as it is much more efficient than full first-order theorem-proving. We could in principle make additional inferences with theorem-proving, such as answering *What did Google buy?* from *Google bought the largest video website* and *YouTube is the largest video website.*

System	Answers	Correct
Relational-LDA	7046	11.6%
Reverb	180	89.4%
CCG-Baseline	203	95.8%
CCG-WordNet	211	94.8%
CCG-Distributional@250	250	94.1%
CCG-Distributional@500	500	82.0%

Table 2: Results on wide-coverage Question Answering task. CCG-Distributional ranks question/answer pairs by confidence—@250 means we evaluate the top 250 of these. It is not possible to give a recall figure, as the total number of correct answers in the corpus is unknown.

are in the same cluster, marginalizing over their argument types. Answers are ranked by this probability. For CCG-WordNet, we check if the question predicate is a hypernym of the candidate answer predicate (using any WordNet sense of either term).

Results are shown in Table 2. Relational-LDA induces many meaningful clusters, but predicates must be assigned to one of 100 relations, so results are dominated by large, noisy clusters (it is not possible to take the N-best answers as the cluster assignments do not have a confidence score). The CCG-Baseline errors are mainly caused by parser errors, or relations in the scope of non-factive operators. CCG-WordNet adds few answers to CCG-Baseline, reflecting the limitations of hand-built ontologies.

CCG-Distributional substantially improves recall over other approaches whilst retaining good precision, demonstrating that we have learnt a powerful model of lexical semantics. Table 3 shows some correctly answered questions. The system improves over the baseline by mapping expressions such as *merge with* and *acquisition of* to the same relation cluster. Many of the errors are caused by conflating predicates where the entailment only holds in one direction, such as *was elected to* with *ran for*. Hierarchical clustering could be used to address this.

8.2 Experiments on the FraCaS Suite

We are also interested in evaluating our approach as a model of formal semantics—demonstrating that it is possible to integrate the formal semantics of Steedman (2012) with our distributional clusters. The FraCaS suite (Cooper et al., 1996)[11] contains a hand-built set of entailment problems designed to be challenging in terms of formal semantics. We use Section 1, which contains 74 problems requiring an understanding of quantifiers[12]. They do not require any knowledge of lexical semantics, meaning we can evaluate the formal component of our system in isolation. However, we use the same representations as in our previous experiment, even though the clusters provide no benefit on this task. Figure 5 gives an example problem.

The only previous work we are aware of on this dataset is by MacCartney and Manning (2007). This approach learns the monotonicity properties of words from a hand-built training set, and uses this to transform a sentence into a polarity annotated string. The system then aims to transform the premise string into a hypothesis. Positively polarized words can be replaced with less specific ones (e.g. by deleting adjuncts), whereas negatively polarized words can be replaced with more specific ones (e.g. by adding adjuncts). Whilst this is high-precision and often useful, this logic is unable to perform inferences with multiple premise sentences (in contrast to our first-order logic).

Development consists of adding entries to our lexicon for quantifiers. For simplicity, we treat multi-word quantifiers like *at least a few*, as being multi-word expressions—although a more compositional analysis may be possible. Following MacCartney and Manning (2007), we do not use held-out data—each problem is designed to test a different issue, so it is not possible to generalize from one subset of the suite to another. As we are interested in evaluating the semantics, not the parser, we manually supply gold-standard lexical categories for sentences with parser errors (any syntactic mistake causes incorrect semantics). Our derivations produce a distribution over logical forms—we license the inference if it holds in any interpretation with non-zero probability. We use the Prover9 (McCune, 2005) theorem prover for inference, returning *yes* if the premise implies the hypothesis, *no* if it implies the negation of the hypothesis, and *unknown* otherwise.

Results are shown in Table 4. Our system im-

[11]We use the version converted to machine readable format by MacCartney and Manning (2007)

[12]Excluding 6 problems without a defined solution.

Question	Answer	Sentence
What did Delta merge with?	Northwest	The 747 freighters came with Delta's acquisition of Northwest
What spoke with Hu Jintao?	Obama	Obama conveyed his respect for the Dalai Lama to China's president Hu Jintao during their first meeting...
What arrived in Colorado?	Zazi	Zazi flew back to Colorado...
What ran for Congress?	Young	... Young was elected to Congress in 1972

Table 3: Example questions correctly answered by CCG-Distributional.

Premises:	Every European has the right to live in Europe. Every European is a person. Every person who has the right to live in Europe can travel freely within Europe.
Hypothesis:	Every European can travel freely within Europe
Solution:	Yes

Figure 5: Example problem from the FraCaS suite.

System	**Single Premise**	**Multiple Premises**
MacCartney&Manning 07	84%	-
MacCartney&Manning 08	98%	-
CCG-Dist (parser syntax)	70%	50%
CCG-Dist (gold syntax)	89%	80%

Table 4: Accuracy on Section 1 of the FraCaS suite. Problems are divided into those with one premise sentence (44) and those with multiple premises (30).

proves on previous work by making multi-sentence inferences. Causes of errors include missing a distinct lexical entry for plural *the*, only taking existential interpretations of bare plurals, failing to interpret mass-noun determiners such as *a lot of*, and not providing a good semantics for non-monotone determiners such as *most*. We believe these problems will be surmountable with more work. Almost all errors are due to incorrectly predicting *unknown* — the system makes just one error on *yes* or *no* predictions (with or without gold syntax). This suggests that making first-order logic inferences in applications will not harm precision. We are less robust than MacCartney and Manning (2007) to syntax errors but, conversely, we are able to attempt more of the problems (i.e. those with multi-sentence premises). Other approaches based on distributional semantics seem unable to tackle any of these problems, as they do not represent quantifiers or negation.

9 Related Work

Much work on semantics has taken place in a supervised setting—for example the GeoQuery (Zelle and Mooney, 1996) and ATIS (Dahl et al., 1994) semantic parsing tasks. This approach makes sense for generating queries for a specific database, but means the semantic representations do not generalize to other datasets. There have been several attempts to annotate larger corpora with semantics—such as Ontonotes (Hovy et al., 2006) or the Groningen Meaning Bank (Basile et al., 2012). These typically map words onto senses in ontologies such as WordNet, VerbNet (Kipper et al., 2000) and FrameNet (Baker et al., 1998). However, limitations of these ontologies mean that they do not support inferences such as *X is the author of Y* $\rightarrow$ *X wrote Y*.

Given the difficulty of annotating large amounts of text with semantics, various approaches have attempted to learn meaning without annotated text. Distant Supervision approaches leverage existing knowledge bases, such as Freebase (Bollacker et al., 2008), to learn semantics (Mintz et al., 2009; Krishnamurthy and Mitchell, 2012). Dependency-based Compositional Semantics (Liang et al., 2011) learns the meaning of questions by using their answers as denotations—but this appears to be specific to question parsing. Such approaches can only learn the pre-specified relations in the knowledge base.

The approaches discussed so far in this section have all attempted to map language onto some pre-specified set of relations. Various attempts have been made to instead induce relations from text by clustering predicates based on their arguments. For example, Yao et al. (2011) propose a series of LDA-based models which cluster relations between entities based on a variety of lexical, syntactic and semantic features. Unsupervised Semantic Parsing (Poon and Domingos, 2009) recursively clusters fragments of dependency trees based on their arguments. Although USP is an elegant model, it is too computationally expensive to run on large corpora. It is also based on frame semantics, so does not cluster equivalent predicates with different frames. To our knowledge, our work is the first such approach to be integrated within a linguistic theory supporting formal semantics for logical operators.

Vector space models represent words by vectors based on co-occurrence counts. Recent work has tackled the problem of composing these matrices to build up the semantics of phrases or sentences (Mitchell and Lapata, 2008). Another strand (Coecke et al., 2010; Grefenstette et al., 2011) has shown how to represent meanings as tensors, whose order depends on the syntactic category, allowing an elegant correspondence between syntactic and semantic types. Socher et al. (2012) train a composition function using a neural network—however their method requires annotated data. It is also not obvious how to represent logical relations such as quantification in vector-space models. Baroni et al. (2012) make progress towards this by learning a classifier that can recognise entailments such as *all dogs* $\implies$ *some dogs*, but this remains some way from the power of first-order theorem proving of the kind required by the problem in Figure 5.

An alternative strand of research has attempted to build computational models of linguistic theories based on formal compositional semantics, such as the CCG-based Boxer (Bos, 2008) and the LFG-based XLE (Bobrow et al., 2007). Such approaches convert parser output into formal semantic representations, and have demonstrated some ability to model complex phenomena such as negation. For lexical semantics, they typically compile lexical resources such as VerbNet and WordNet into inference rules—but still achieve only low recall on open-domain tasks, such as RTE, mostly due to the low coverage of such resources. Garrette et al. (2011) use distributional statistics to determine the probability that a WordNet-derived inference rule is valid in a given context. Our approach differs in that we learn inference rules not present in WordNet. Our lexical semantics is integrated into the lexicon, rather than being implemented as additional inference rules, meaning that inference is more efficient, as equivalent statements have the same logical form.

Natural Logic (MacCartney and Manning, 2007) offers an interesting alternative to symbolic logics, and has been shown to be able to capture complex logical inferences by simply identifying the scope of negation in text. This approach achieves similar precision and much higher recall than Boxer on the RTE task. Their approach also suffers from such limitations as only being able to make inferences between two sentences. It is also sensitive to word order, so cannot make inferences such as *Shakespeare wrote Macbeth* $\implies$ *Macbeth was written by Shakespeare*.

10 Conclusions and Future Work

This is the first work we are aware of that combines a distributionally induced lexicon with formal semantics. Experiments suggest our approach compares favourably with existing work in both areas.

Many potential areas for improvement remain. Hierachical clustering would allow us to capture hypernym relations, rather than the synonyms captured by our flat clustering. There is much potential for integrating existing hand-built resources, such as Ontonotes and WordNet, to improve the accuracy of clustering. There are cases where the existing CCGBank grammar does not match the required predicate-argument structure—for example in the case of light verbs. It may be possible to rebank CCGBank, in a way similar to Honnibal et al. (2010), to improve it on this point.

Acknowledgements

We thank Christos Christodoulopoulos, Tejaswini Deoskar, Mark Granroth-Wilding, Ewan Klein, Katrin Erk, Johan Bos and the anonymous reviewers for their helpful comments, and Limin Yao for sharing code. This work was funded by ERC Advanced Fellowship 249520 GRAMPLUS and IP EC-FP7-270273 Xperience.

References

C.F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

M. Baroni, R. Bernardi, N.Q. Do, and C. Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of EACL*, pages 23–32. Citeseer.

V. Basile, J. Bos, K. Evang, and N. Venhuizen. 2012. Developing a large semantically annotated corpus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC12). To appear*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 610–619. Association for Computational Linguistics.

C. Biemann. 2006. Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80. Association for Computational Linguistics.

D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

D. G. Bobrow, C. Condoravdi, R. Crouch, V. De Paiva, L. Karttunen, T. H. King, R. Nairn, L. Price, and A. Zaenen. 2007. Precision-focused textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 16–21. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

J. Bos and K. Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635. Association for Computational Linguistics.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

N. Chinchor and P. Robinson. 1997. Muc-7 named entity task definition. In *Proceedings of the 7th Conference on Message Understanding*.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04. Association for Computational Linguistics.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis: A Festschrift for Joachim Lambek*, 36(1-4):345–384.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. *FraCaS Deliverable D*, 16.

Ido Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL recognising textual entailment challenge. *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190.

D.A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

Hal Daumé III. 2007. Fast search for dirichlet process mixture models. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, San Juan, Puerto Rico.

D. Davidson. 1967. 6. the logical form of action sentences. *Essays on actions and events*, 1(9):105–149.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1535–1545. Association for Computational Linguistics.

T. Fountain and M. Lapata. 2011. Incremental models of natural language category acquisition. In *Proceedings of the 32st Annual Conference of the Cognitive Science Society*.

D. Garrette, K. Erk, and R. Mooney. 2011. Integrating logical representations with probabilistic information using markov logic. In *Proceedings of the Ninth International Conference on Computational Semantics*, pages 105–114. Association for Computational Linguistics.

D. Graff, J. Kong, K. Chen, and K. Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2011. Concrete sentence spaces for compositional distributional models of meaning. *Computational Semantics IWCS 2011*, page 125.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

M. Honnibal, J.R. Curran, and J. Bos. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

P. Kingsbury and M. Palmer. 2002. From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*, pages 1989–1993. Citeseer.

K. Kipper, H.T. Dang, and M. Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the National Conference on Artificial Intelligence*, pages 691–696. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 754–765. Association for Computational Linguistics.

P. Liang, M.I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Proc. Association for Computational Linguistics (ACL)*.

Dekang Lin and Patrick Pantel. 2001. DIRT - Discovery of Inference Rules from Text. In *In Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Bill MacCartney and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, RTE '07, pages 193–200. Association for Computational Linguistics.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

W. McCune. 2005. Prover9 and Mace4. `http://cs.unm.edu/~mccune/mace4/`.

G.A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics.

J. Mitchell and M. Lapata. 2008. Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, pages 236–244.

R.M. Neal. 2000. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

D.O. O'Seaghdha. 2010. Latent variable models of selectional preference. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 435–444. Association for Computational Linguistics.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 1–10. Association for Computational Linguistics.

Hoifung Poon and Pedro Domingos. 2010. Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 296–305. Association for Computational Linguistics.

A. Ritter, O. Etzioni, et al. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434. Association for Computational Linguistics.

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1088–1098. Association for Computational Linguistics.

R. Socher, B. Huval, C.D. Manning, and A.Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Mark Steedman. 2012. *Taking Scope: The Natural Semantics of Quantifiers*. MIT Press.

Idan Szpektor, Eyal Shnarch, and Ido Dagan. 2007. Instance-based evaluation of entailment rule acquisition. In *In Proceedings of ACL 2007*, volume 45, page 456.

Ivan Titov and Alexandre Klementiev. 2011. A bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1445–1455, Portland, Oregon, USA, June. Association for Computational Linguistics.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1456–1466. Association for Computational Linguistics.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2012. Unsupervised relation discovery with sense disambiguation. In *ACL (1)*, pages 712–720.

J.M. Zelle and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055.