Claudio Delli Bovi

SAPIENZA
UNIVERSITÀ DI ROMA

Combined Distributional and Logical Semantics

M. Lewis *M. Steedman*

Introduction

Mapping *natural language* to *meaning representations* is a tough challenge of NLP which requires knowledge of language at many different levels.

For instance, consider what is needed to answer a question like

Did Google buy YouTube?

from the following sentences:

- *Google purchased YouTube*
- *Google's acquisition of YouTube*
- *Google acquired every company*
- *YouTube may be sold to Google*
- *Google will buy YouTube or Microsoft*
- *Google didn't takeover YouTube*

Introduction

Mapping *natural language* to *meaning representations* is a tough challenge of NLP which requires knowledge of language at many different levels.

For instance, consider what is needed to answer a question like

Did Google buy YouTube?

from the following sentences:

- *Google purchased YouTube*
- *Google's acquisition of YouTube*
- *Google acquired every company*
- *YouTube may be sold to Google*
- *Google will buy YouTube or Microsoft*
- *Google didn't takeover YouTube*

knowledge of *lexical semantics* (*buy* and *purchase* as synonyms)

interpretation of *quantifiers*, *negatives*, *modals* and *disjunction* (*every*, *may*, *or*, *didn't*)

At a glance

Two approaches so far:

- **Distributional semantics**, in which the meaning of a word is induced from its usage in large corpora

✓ successful in modeling the meanings of content words
✓ unsupervised: no dependence on hand-built training data
× less clear how to apply on function words and operators

- **Formal semantics**, i.e. computational models based on a formal logical description

✓ operators and function words are naturally expressed
✓ powerful engines available for reasoning and inference
× low recall on practical applications (reliance on training data)

At a glance

None of the two seems to be enough to accomplish the task…

The idea: take the best of both worlds!

- Follow *formal semantics* in mapping language to logical representations;
- Induce relational constants by *offline distributional clustering* at the level of predicate-argument structure.

At a glance

In the following...

- Background: Combinatory Categorial Grammars (CCGs)
- Overview of the approach
- Parsing and initial semantic analysis
- Entity typing model
- Distributional semantic analysis
- Cross-lingual cluster alignment
- Experiments: Q&A and Machine Translation

Background: CCGs

Combinatory Categorial Grammar (CCG) is a strongly lexicalized theory of language in which *lexical entries for words contain all language-specific information*.

For each word, the associated lexical entry contains:

- a *syntactic category,* which determines which other categories the word may combine with;
- a *semantic interpretation*, which defines the related compositional semantics.

For example, a possible entry in the lexicon could be:

$$\boxed{write} \vdash \boxed{(S\backslash NP)/NP \;\; \lambda y.\lambda x.\mathrm{write}'(x, y)}$$

Lexeme CCG category

Background: CCGs

Lexeme — CCG category

write ⊢ $(S\backslash NP)/NP$ λy.λx.write‘(x, y)

Syntax — Semantics

The so-called *Lambek notation* (argument under slash) reads like this:

- A/B = "give me a B to my **right**, then I'll give you an A"
- A\B = "give me a B to my **left**, then I'll give you an A"

λ-calculus expression paired with the syntactic type: *syntactic* and *semantic* information captured *jointly*

CCG parsing: a toy example

CCG is fun

CCG parsing: a toy example

$$\frac{\text{CCG}}{\begin{matrix} NP \\ CCG \end{matrix}} \qquad \frac{\text{is}}{\begin{matrix} S\backslash NP/ADJ \\ \lambda f.\lambda x.f(x) \end{matrix}} \qquad \frac{\text{fun}}{\begin{matrix} ADJ \\ \lambda x.fun(x) \end{matrix}}$$

First, use the *lexicon* to match words and phrases with their categories

CCG parsing: a toy example

$$\dfrac{\text{CCG}}{\begin{matrix} NP \\ CCG \end{matrix}} \qquad \dfrac{\text{is}}{\begin{matrix} S\backslash NP/ADJ \\ \lambda f.\lambda x.f(x) \end{matrix}} \qquad \dfrac{\text{fun}}{\begin{matrix} ADJ \\ \lambda x.fun(x) \end{matrix}}$$

$$\xrightarrow{\hspace{6cm}} >$$

$$\begin{matrix} S\backslash NP \\ \lambda x.fun(x) \end{matrix}$$

Forward Function Application:

A/B: f **B:** a ⇒ **A:** f(a)

CCG parsing: a toy example

$$
\begin{array}{ccc}
\text{CCG} & \text{is} & \text{fun} \\
\hline
NP & S\backslash NP/ADJ & ADJ \\
CCG & \lambda f.\lambda x.f(x) & \lambda x.fun(x) \\
\end{array}
$$

$$
\dfrac{S\backslash NP/ADJ : \lambda f.\lambda x.f(x) \quad ADJ : \lambda x.fun(x)}{S\backslash NP : \lambda x.fun(x)} >
$$

$$
\dfrac{NP : CCG \quad S\backslash NP : \lambda x.fun(x)}{S : fun(CCG)} <
$$

Backward Function Application:

B: a **A\B:** f ⇒ **A:** f(a)

Approach overview

The proposed approach uses a CCG parser to map natural language sentences to *first-order logic representations*, where the meaning of *content words* is modeled using *distributional statistics*.

Non-logical symbols (e.g. *write*) stand for *arbitrary relation identifiers* (e.g. *relation37*) connected to distributional clusters at the level of *predicate-argument structure*.

Input Sentence

Shakespeare wrote Macbeth

$\Downarrow$

Intial semantic analysis

$write_{arg0,arg1}(shakespeare, macbeth)$

$\Downarrow$

Entity Typing

$write_{arg0:PER,arg1:BOOK}(shakespeare{:}PER, macbeth{:}BOOK)$

$\Downarrow$

Distributional semantic analysis

$relation37(shakespeare{:}PER, macbeth{:}BOOK)$

Approach overview

The CCG lexicon is first mapped to a *deterministic logical form* (predicates)

A *typing model* is then built into the derivation: all terms denoting entities are further subcategorized with a more detailed type

Predicates are finally *clustered* based on their arguments

Input Sentence

Shakespeare wrote Macbeth

$\Downarrow$

Intial semantic analysis

$write_{arg0,arg1}(shakespeare, macbeth)$

$\Downarrow$

Entity Typing

$write_{arg0:PER,arg1:BOOK}(shakespeare{:}PER, macbeth{:}BOOK)$

$\Downarrow$

Distributional semantic analysis

$relation37(shakespeare{:}PER, macbeth{:}BOOK)$

Initial semantic analysis

The *initial semantic analysis* comprises three steps:

- **Syntactic parsing** (as shown before) with the C&C CCG parser trained *on CCGBank* (a translation of the Penn Treebank into a corpus of CCG derivations) yielding *POS tags* and *syntactic categories*;

Initial semantic analysis

The *initial semantic analysis* comprises three steps:

- **Syntactic parsing** (as shown before) with the *C&C CCG parser* trained *on CCGBank* (a translation of the Penn Treebank into a corpus of CCG derivations) yielding *POS tags* and *syntactic categories*;
- **Mapping** from *parser output* to *logical form* (automatic);

$$\text{author} \Rightarrow N/PP[of] \quad \lambda x \lambda y.author_{arg0,argOf}(y,x)$$

$$\text{write} \Rightarrow (S\backslash NP)/NP \quad \lambda x \lambda y.write_{arg0,arg1}(y,x)$$

Initial semantic analysis

The *initial semantic analysis* comprises three steps:

- **Syntactic parsing** (as shown before) with the *C&C CCG parser* trained *on CCGBank* (a translation of the Penn Treebank into a corpus of CCG derivations) yielding *POS tags* and *syntactic categories*;
- **Mapping** from *parser output* to *logical form* (automatic);

author	$N/PP[of]$	$\lambda x \lambda y.author_{arg0,argOf}(y,x)$
write	$(S\backslash NP)/NP$	$\lambda x \lambda y.write_{arg0,arg1}(y,x)$

- A few **manually-added entries** for critical closed-class function words like *negatives* and *quantifiers*;

every	$NP^{\uparrow}/N$	$\lambda p \lambda q.\forall x[p(x) \rightarrow q(x)]$
not	$(S\backslash NP)/(S\backslash NP)$	$\lambda p \lambda x.\neg p(x)$

Entity typing model

Aim: cluster entities based on the *noun* and *unary predicate* applied to them. Non-trivial, as predicates and arguments can be *ambiguous* between *multiple types*, e.g.

pair $(born_{argIN}, 1961)$	should map to	DAT *type*
pair $(born_{argIN},$ Hawaii)	should map to	LOC *type*

Key assumption: in a predicate, the type of each argument depends *only* on the predicate itself and its arguments.

Topic modeling based on standard **Latent Dirichlet Allocation** (LDA): assign each type *j* a *multinomial distribution* ϕ_j over arguments and each unary predicate *i* a *multinomial distribution* θ_i over topics, then construct a document for each *unary predicate*, based on all of its argument entities.

Entity typing model

Typing in logical form: all constants and variables representing *entities* x can be assigned a *distribution over types* $p_x(t)$ using the type model.

Such distributions are *updated as the LDA process goes on*, and then used to overcome lexical ambiguity during the derivation. For instance, consider the word *suit* in the following parse: *to file a suit*

$$\frac{\text{file}}{(S\backslash NP)/NP \qquad \lambda y{:}\left\{\begin{array}{c} DOC = 0.5 \\ LEGAL = 0.4 \\ CLOTHES = 0.01 \\ \ldots \end{array}\right\} \lambda x{:}\left\{\begin{array}{c} PER = 0.7 \\ ORG = 0.2 \\ \ldots \end{array}\right\}.file_{arg0,arg1}(x,y)}$$

$$\frac{\text{a suit}}{NP^{\uparrow} \qquad \lambda p.\exists y{:}\left\{\begin{array}{c} CLOTHES = 0.6 \\ LEGAL = 0.3 \\ DOC = 0.001 \\ \ldots \end{array}\right\}[suit'(y) \wedge p(y)]}$$

suit as a piece of clothing
or
suit as a civil proceeding?

Entity typing model

Typing in logical form: all constants and variables representing *entities* x can be assigned a *distribution over types* $p_x(t)$ using the type model.

Such distributions are *updated as the LDA process goes on*, and then used to overcome lexical ambiguity during the derivation. For instance, consider the word *suit* in the following parse: *to file a suit*

$$\frac{\text{file} \quad \text{a} \quad \text{suit}}{S\backslash NP}<$$

$$\lambda x: \left\{ \begin{array}{c} PER = 0.7 \\ ORG = 0.2 \\ \ldots \end{array} \right\} \exists y: \left\{ \begin{array}{c} LEGAL = 0.94 \\ CLOTHES = 0.05 \\ DOC = 0.004 \\ \ldots \end{array} \right\})[suit'(y) \wedge file_{arg0,arg1}(x,y)]$$

what if we had a parse like *to wear a suit*?

Distributional semantic analysis

Typed binary predicates are grouped into clusters, ech representing a *distinct semantic relation.* Clusters are built on the expected number of times a predicate holds between each argument pair in the corpus.

⇒ a predicate like $\boldsymbol{write}\ (\boldsymbol{PER}, \boldsymbol{BOOK})$ may contain non-zero counts for entity-pairs such as $(Shakespeare, Macbeth)$, $(Dickens, Oliver\ Twist)$ and so on...

⇒ $\boldsymbol{author}\ (\boldsymbol{PER}, \boldsymbol{BOOK})$ and $\boldsymbol{write}\ (\boldsymbol{PER}, \boldsymbol{BOOK})$ are likely to have similar counts, while predicates like $\boldsymbol{bornIn}\ (\boldsymbol{PER}, \boldsymbol{LOC})$ and $\boldsymbol{bornIn}(\boldsymbol{PER}, \boldsymbol{DAT})$ will cluster separately, despite the ambiguity at the lexical level.

Distributional semantic analysis

Many algorithms can be used to effectively cluster predicates wrt their arguments, as long as they are *scalable* to a very large number of predicates and (possibly) *non-parametric*.

A suitable choice is the simple yet very efficient **Chinese Whispers Algorithm (CWA)**. It goes as follows:

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

Semantic parsing with relation clusters

The final step is to use the computed relation clusters in the lexical entries of the CCG semantic derivation.

A *packed logical form* is produced, capturing the full distribution of types over logical forms and making the predicate a function from *argument types* to *relations*:

$$born \vdash (S \backslash NP)/PP[in]:$$
$$\lambda y \lambda x. \left\{ \begin{array}{l} (x: PER, y: LOC) \Rightarrow rel49 \\ \underbrace{(x: PER, y: DAT)}_{\text{Argument types}} \Rightarrow \underbrace{rel53}_{\text{Distributional clusters}} \end{array} \right\} (x, y)$$

Semantic parsing with relation clusters

Distributions over argument types then imply a *distribution over relations*.

As an example, consider the two *argument pairs* $(Obama, Hawaii)$ and $(Obama, 1961)$ and the following *type distributions*:

- $Obama/ob$: ($PER = 0.9,\ LOC = 0.1$) ;
- $Hawaii/hw$: ($LOC = 0.7,\ DAT = 0.3$) ;
- $1961/1961$: ($LOC = 0.1,\ DAT = 0.9$) ;

The output *packed logical form* will be:

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

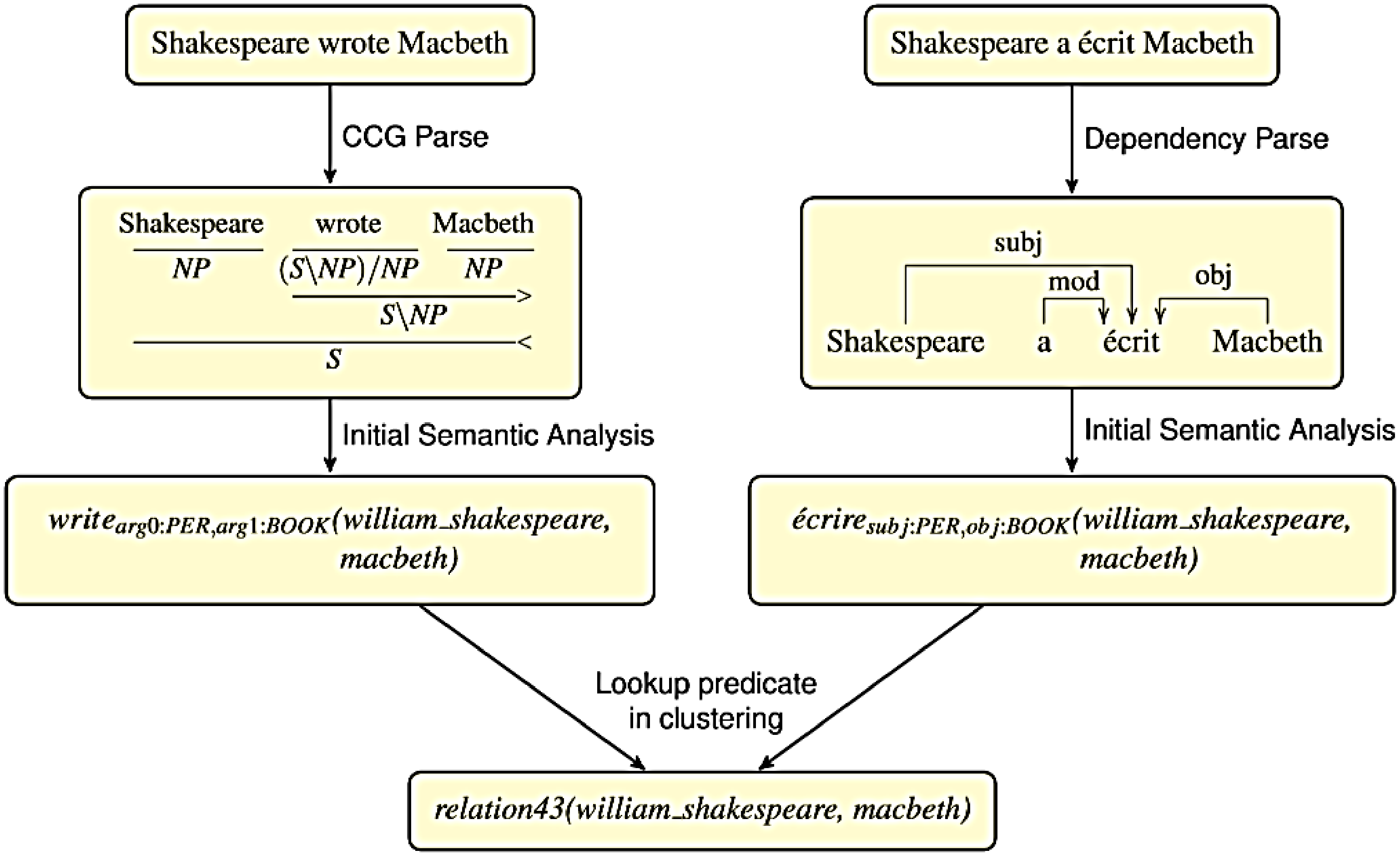
Is it language-independent?

Idea: the problem of learning binary relations between entities could be generalized by treating a foreign expression *as a paraphrase for an English expression*.

How?
In principle, the clusters obtained with the proposed approach can be treated as *language-indepedent (interlingua) semantic relations*, just by mapping clustered expressions in different languages onto the same relation.

⇒ No (or little) parallel corpora needed in a hypothetical implementation for Machine Translation: alignment at the entity-level is exploited!

Is it language-independent?



Cross-lingual cluster alignment

The process is carried out in the same way as before: we end up with a set of *monolingual relation clusters* as a result of the CWA.

In order to find an **alignment** between such clusters in different languages, a *simple greedy procedure* is used: entity-pair vectors for each predicate in a relation cluster are merged and, for those occurring in both languages, a *cosine similarity* measure is computed.

1. Initialize the *alignment* $A \longleftarrow \{\,\}$;
2. **while** $R_{L1} \neq \{\,\} \wedge R_{L2} \neq \{\,\}$ **do**

$$(r_1,\ r_2) \longleftarrow \underset{(r_1, r_2)\, \in R_{L1} \times R_{L2}}{\arg\max}\ sim(r_1,\ r_2)\ ;$$

$$A \longleftarrow A\ \cup \{\, (r_1,\ r_2) \,\}\ ;$$
$$R_{L1} \longleftarrow R_{L1} \,/\, \{\, r_1 \,\}\ ;$$
$$R_{L2} \longleftarrow R_{L2} \,/\, \{\, r_2 \,\}\ ;$$

Experiment #1: Cross-lingual Q&A task

A first evaluation of the proposed approach is based on a *cross-lingual question answering task*, where a question is asked in language L and then answered by the system from a corpus of language L'.

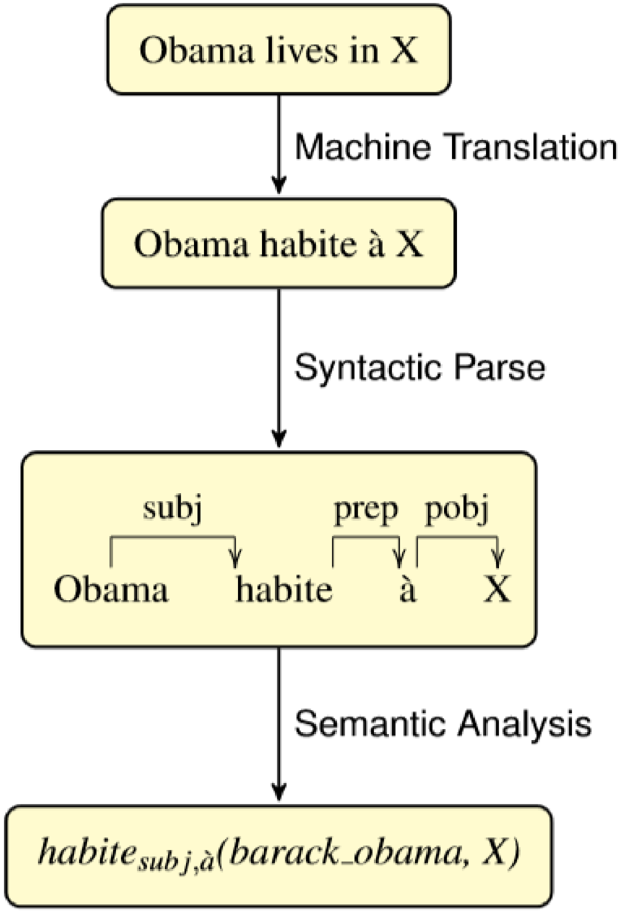
To assess performances, human annotators are shown *question*, *answer entity*, and *sentence that provided the answer*. They are then asked whether the answer is a reasonable conclusion based on the sentence.

Question	Answer
X wins the FA Cup	**Portsmouth FC** remporte la FA Challenge Cup en s'imposant en finale face à Wolverhampton Wanderers FC
X is a band from Finland	**Yearning** est un groupe Finlande de doom metal atmosphérique
X bat Kurt Angle	**Anderson** defeated Kurt Angle and Abyss to advance to the finals
X est une ville de Kirghizistan	**Il'chibay** is a village in the Issyk Kul Province of Kyrgyzstan

Experiment #1: Cross-lingual Q&A task

The system attempts the task by mapping both *question* and *candidate answer* sentences on to a logical form using its *relation clusters*: then it determines whether they express they same relation.

A baseline is provided by a *Moses* model trained on the *Europarl corpus*.
To accomplish the task, the question is first *translated* from language L to L' taking the 50-best translations; these are then parsed to extract a set of patterns, which are used to find candidate answers.



Experiment #1: Cross-lingual Q&A task

English→ French	Answers	Correct
Baseline	269	86%
Clusters (best 270)	270	100%
Clusters (all)	1032	72%

French→ English	Answers	Correct
Baseline	274	85%
Clusters (all)	401	93%

Best-N results are shown to illustrate the accuracy of the cluster-based system at the same rank as the baseline.

Languages: English, French

Corpora: Wikipedia

English corpus:
POS e CCG tags provided by the C&C parser (trained on CCGBanks).

French corpus:
Tags and parses provided by MElt and Malt Parser (trained on the French Treebank).

Experiment #2: Translation reranking

The second experiment investigates the possibility of *reranking the output of a machine translation system*, on the basis of whether the *semantic parse of the source sentence is consistent* with that of candidate translations.

A sample of French sentences (for which a semantic parse can be produced) are translated to English using Moses, and then parsed again:

- If the semantic parse for the best translation does **not** match the source parse, an alternative is selected from the 50-best list (so to have the most closely matched parses);
- Otherwise the sentence is discarded from the evaluation, as the two systems agree on the semantics.

Experiment #2: Translation reranking

Human annotators were asked to assess the reranking performance by examining (in a random order) the *best translation* and the *translation chosen by the re-ranker* against the source sentence.

	Percentage of translations preferred
1-best Moses translation	5%
Cluster-based Reranker	39%
No preference	56%

No preference expressed: mostly due to syntax errors in the translation!

Total number of evaluated sentences: 87

References

- M. Lewis, M. Steedman, *Combined Distributional and Logical Semantics.* In *Transactions of the Association for Computational Linguistics 1*, pp. 179-192, 2013.

- M. Lewis, M. Steedman, *Unsupervised Induction of Cross-lingual Semantic Relations.* In *Proceedings of the Conference on Empirical Methods in NLP*, pp. 681–692, 2013.

C&C CCG parser + other tools

`http://svn.ask.it.usyd.edu.au/trac/candc`

CCGBank and **CCG-related software**

`http://groups.inf.ed.ac.uk/ccg`

OpenCCG:
`http://openccg.sourceforge.net`

Additional readings

- M. Steedman, J. Baldridge, *Combinatory Categorial Grammars*.
  In R. Borsley and K. Borjars (eds.), *Non-Transformational Syntax*, Blackwell, 2005.

- D. M. Blei, A. Y. Ng, M. I. Jordan, *Latent Dirichlet Allocation.*
  In *Journal of Machine Learning Research 3*, pp. 993-1022, 2003.

- C. Biemann, *Chinese Whispers, an Efficient Graph Clustering Algorithm and its Application to NLP problems*.
  In *Workshop on TextGraphs at HLT-NAACL*,, pp. 73-80, 2006.