Trial Data

The trail data can be downloaded from the following location.

Trial Data

Training and Development Data

The training data can be downloaded from the following location. In order to use this data, you would need to obtain the OntoNotes v4.0 corpus from LDC. You would have got the information on how to obtain the corpus from LDC when you registered. Since LDC owns the copyright, the files we provide here are semi-offset annotations. You would need to generate the word column in the CoNLL format file (.conll) which we have one per document, using the information below:

- Training Data:
- conll-2011-train.v2.tar.gz

- Development Data:
- conll-2011-dev.v2.tar.gz

Test Data

The test data can be downloaded from the following location. Since one of the genre — telephone conversations (tc) is not available in the OntoNotes 4.0 release, so we have made available *_conll files. All other genre files are the usual *_skel which you would have to convert to *_conll using the OntoNotes 4.0 corpus as in the case of the training and development data. Unlike the training and development data, this set does not contain *_gold_skel files, but only *_auto_skel files. The last column containing coreference information is set to "-".

- Test Data:
- Official:
- conll-2011-test-official.v5.tar.gz

- Supplementary (With Gold Mentions and Mention Boundaries):
- conll-2011-test-supplementary.v5.tar.gz

Test Key

The gold key for the above test set can be downloaded from the following location:

- Test Key
- conll-2011-test-key.tar.gz

System Submissions

The system submissions can be downloaded from the link below

- conll-2011-submissions.tar.gz

Steps for putting the data together

- **Unpack the OntoNotes release files obtained from LDC** $ tar -xvM -f LDC2011T03_1.tgz -f LDC2011T03_2.tgz -f LDC2011T03_3.tgz -f LDC2011T03_4.tgz $ tar xvf LDC2011T03.tgz The OntoNotes download from LDC consists of four .tgz files which comprise a multi-part archive. If you try to untar them individually it won't work. Please use the above commands to re-create the archive and then untar it. This has been tested with GNU tar v1.17. Another thing to note is that although the extension implies a gzipped archive, it is not so, and therefore you should not use a -x option.

- **Create the CoNLL format files for each document in the training and development collection**

Once you untar the training and development archives, you will see the following directory structure:

$ tar zxvf conll-2011-train.v0.tar.gz $ tar zxvf conll-2011-dev.v0.tar.gz $ $ cd conll-2011/v0/data/train $ tree -d data/ data/ `-- english `-- annotations |-- bc | |-- cctv | | `-- 00 ... |-- bn | |-- abc | | `-- 00 |-- mz |

`-- sinorama | `-- 10 ... |-- nw | |-- wsj | | |-- 00 ... | `-- xinhua | |-- 00 ... `-- wb |-- a2e | `-- 00 ... 73 directories

This directory tree under data is the same as the one under the ontonotes-release-4.0/data/ directory. Each leaf directory contains files of the form:

[source]_[four-digit-number].[extension]

with six different extensions of the form:

[extension] := [version]_[quality]_[layer] [version] := **v**[number] [quality] := gold|auto [layer] := skel | prop | sense

- **Downlods the scripts**

Download the scripts from the following location

Scripts:

- conll-2011-scripts.v2.tar.gz

Following is the list of all scripts:

scripts/ scripts/skeleton2conll.py scripts/skeleton2conll.sh scripts/conll2coreference.py scripts/conll2coreference.sh scripts/conll2name.py scripts/conll2name.sh scripts/conll2parse.py scripts/conll2parse.sh

First, you have to generate *_conll files from each corresponding *_skel files. The *_skel file is very similar to the *_conll file — it contains information on all the layers of annotation *except* the underlying words. Owing to copyright restrictions on the underlying text, we have to do this workaround. The skeleton2conll.sh shell script is a wrapper for the skeleton2conll.py

script that takes a
*_skel
file as input and generates the corresponding
*_conll
file. The script to get the words back from the trees is non-trivial for the some genre as we have eliminated disfluencies marked by phrases type EDITED in the Treebank. The usage for this script described with an example below:

Usage: **skeleton2conll.sh** -D [ontonotes-release-data-directory] [top-level-directory]

Description
:
[ontonotes-release-data-directory]
: Location of the "data" directory under the OntoNotes 4.0 release
[top-level-directory]
: The directory inside which the
*_skel
files exist and need to be convered to
.conll
files
Example
: The following will create
*_conll
files for all the
*_skel
files in the conll-2011/train directory
skeleton2conll.sh
-D
/nfs/.../ontonotes-release-4.0/data conll-2011/v0/data/train

If you are only going to work using the
*_conll
files, then you don't need to do any further processing after they are generated. But, in case you plan to use the OntoNotes API, it requires individual files for each of the five annotation layers --
.parse, .name, .coref, .prop
and
.sense
(with an optional
[tag]_
prefix). Since the last two in this list occur naturally as standoff annotation, we have included

them as they are in the download. The first three, however, you have to generate using the remaining scripts. As the name suggests each of the python scripts
conll2[layer].py
takes the file with a
*_conll
extension, and produces a
*_[layer]
file. As with the earlier script, for the sake of simplicity, we have provided shell scripts of the same filestem as the python script. These take a directory as their only argument, and traverse all the subdirectories in that directory to create the corresponding layer files in the same directory as the
*_conll
files.

*_conll File Format

The *_conll files contain data in a tabular structure similar to that used by previous CoNLL shared tasks. We are using a [tag]-based extension naming approch where a [tag]
is applied to the
.conll
file to name it, say
.[tag]_conll
. The
[tag]
itself can have multiple components and serves to highlight the characteristics of that
.conll
file. For example, the two tags that we use in the data are "v0_gold" and "v0_auto". Each of it has two (parts separated by underscores). The first one has the same value — "v0" in both cases and indicates the version of the file. The second has two values "gold" and "auto". The "gold" indicates that the annotation is that file is hand-annotated and adjudicated quality, whereas the second means it was produced using a combination of automatic tools. The contents of each of these files comprises of a set of columns. Each column either representing a linear annotation on a sentence, for example, a part of speech annotation which is one part of speech per word, and so one column per layer (in this case part of speech), or there are multiple columns — taken in sync with another column and representing the part that all other words in the sentence play with respect to that word. This is the classic case of predicate argument structure as introduced in the CoNLL-2005 shared task. In this case the number of columns that represent that layer of annotation is variable — one per each predicate. For convenience, we have kept the coreference layer information in the very last column and the predicate argument structure information in a variable number of columns preceeding that. The columns in the
*_conll
file represent the following:

1	Document ID	This is a variation on the document filename
2	Part number	Some files are divided into multiple parts numbere
3	Word number	
4	Word itself	
5	Part-of-Speech	
6	Parse bit	This is the bracketed structure broken before the fi
7	Predicate lemma	The predicate lemma is mentioned for the rows for
8	Predicate Frameset ID	This is the PropBank frameset ID of the predicate i
9	Word sense	This is the word sense of the word in Column 3.
10	Speaker/Author	This is the speaker or author name where availabl
11	Named Entities	These columns identifies the spans representing v
12:N	Predicate Arguments	There is one column each of predicate argument s
N	Coreference	Coreference chain information encoded in a paren

Number and Gender Data

Number and Gender information is one of the core features that any coreference system uses, and therefore, even though it is not directly derived from the OntoNotes data, we are allowing its use in the closed task. However, for the closed task we require that the participants use the same source for extracting number and gender features so that the system results can still be comparable. To this end, we are planning on allowing the use of the number and gender data that was created by **Shane Bergsma** and **Dekang Lin** in the following paper: