Diving for Treasure in a Sea of Scientific Literature

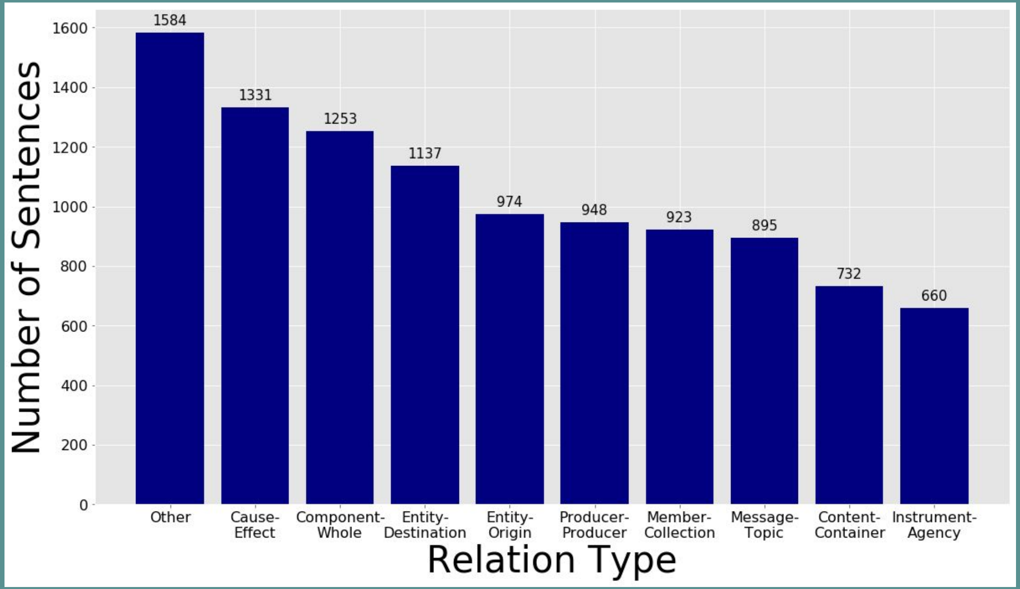
Extracting Scientific Information from Free Text Articles

Aarthi Koripelly
Kyle Chard (Advisor)
Zhi Hong (Advisor)

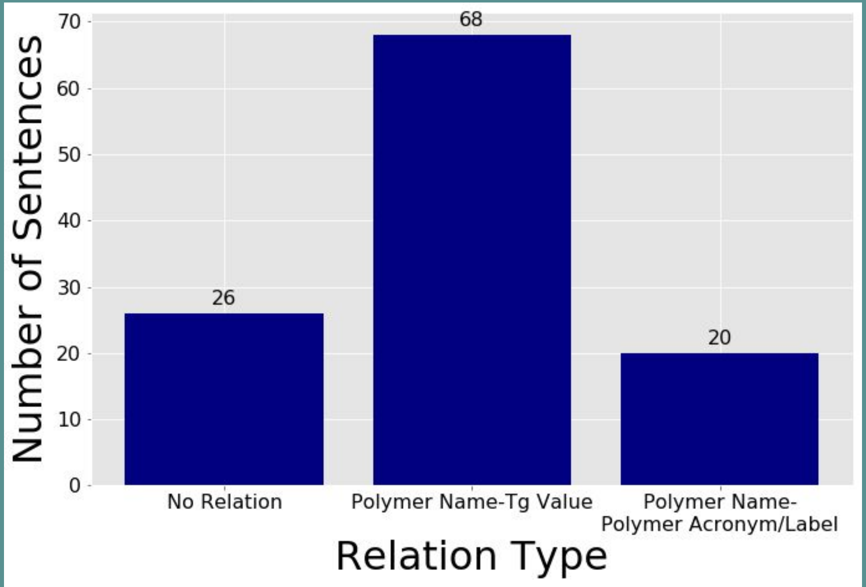
Why?

Dataset Distribution

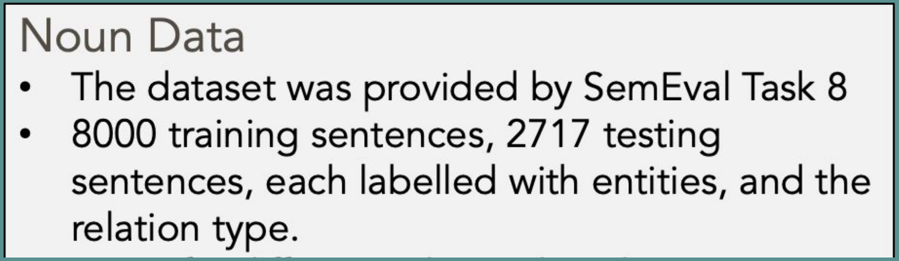
Noun Dataset Distribution



Polymer Dataset Distribution

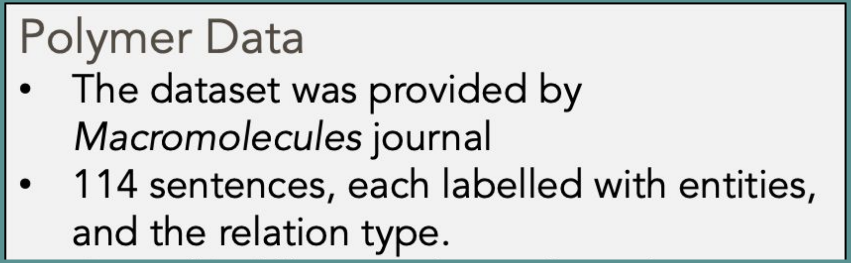


Dataset Distribution



Noun Data

- The dataset was provided by SemEval Task 8
- 8000 training sentences, 2717 testing sentences, each labelled with entities, and the relation type.



Polymer Data

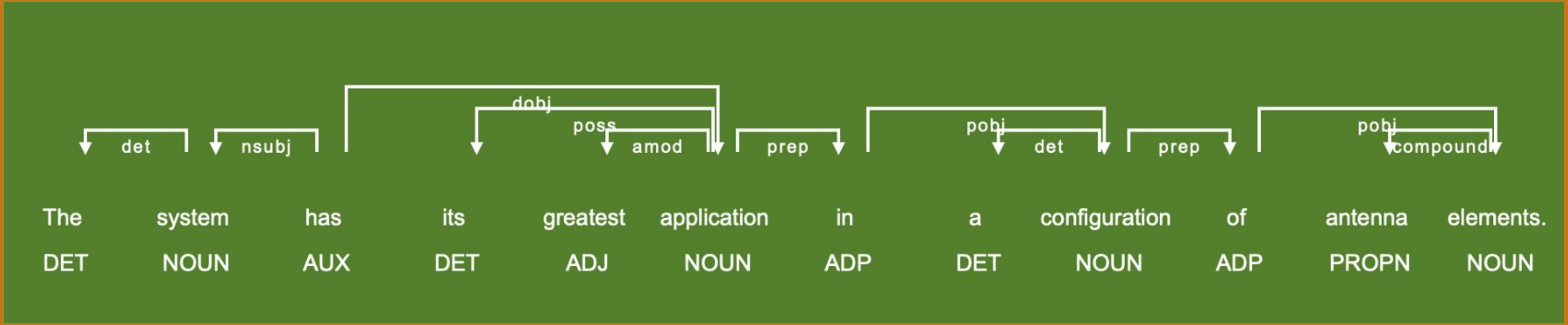
- The dataset was provided by *Macromolecules* journal
- 114 sentences, each labelled with entities, and the relation type.

Example: ‘Wings’ and ‘Fly’
Cause-Effect Relationship

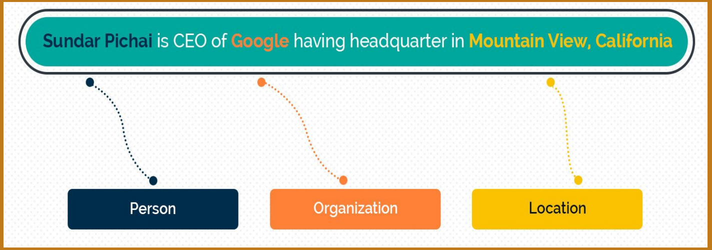
Example: ‘5 7-dibutyl-1 3-dehydroadamantane’ and ‘3’
Polymer Name and Label Relationship

What is a Dependency Parser?

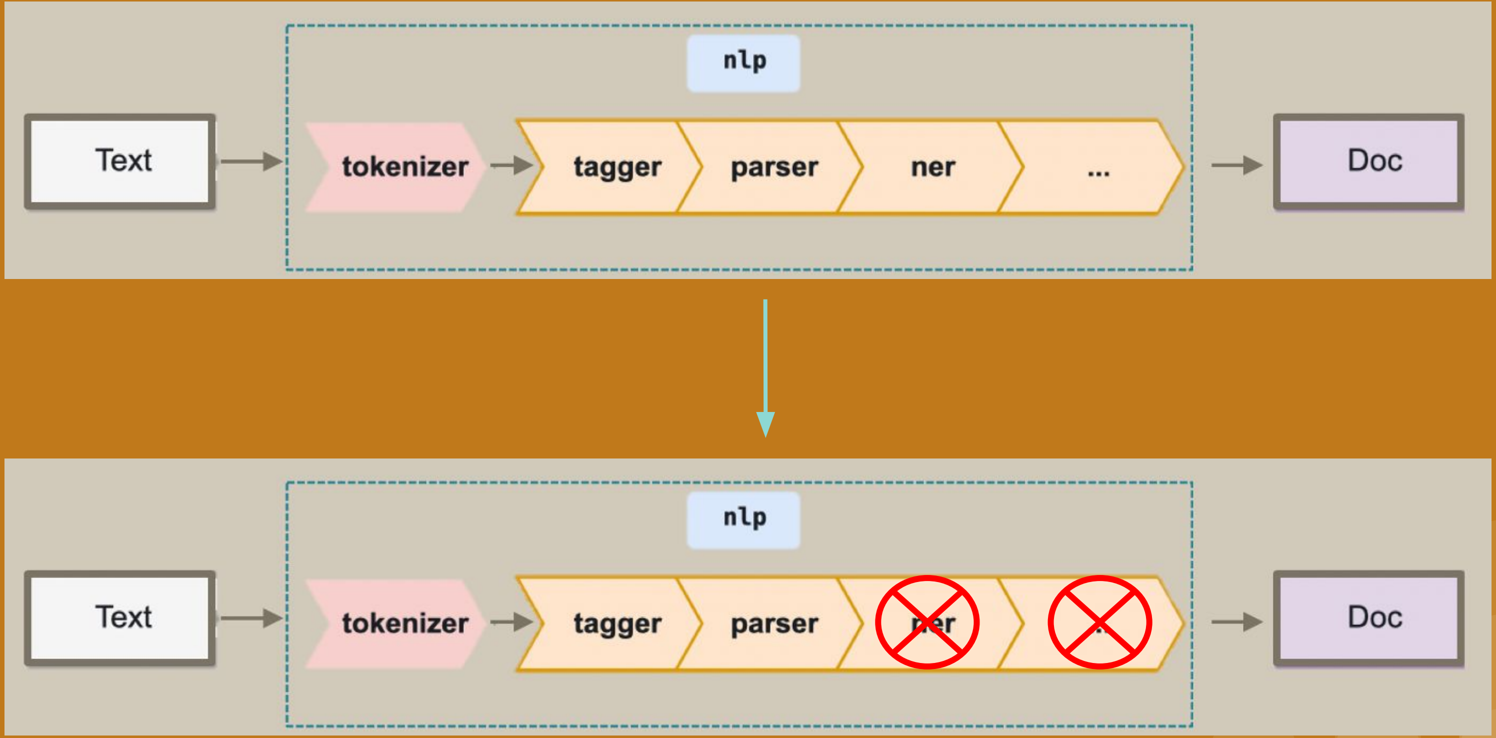
Extracts relationships between words based on parts of speech and semantic relationships



Named Entity Tagger: Extracts relationships between words based on word type (person, place, location, etc)


Sundar Pichai is CEO of Google having headquarter in Mountain View, California
Person
Organization
Location

spaCy NLP Pipeline



Text

nlp

tokenizer

tagger

parser

ner

...

Doc

Text

nlp

tokenizer

tagger

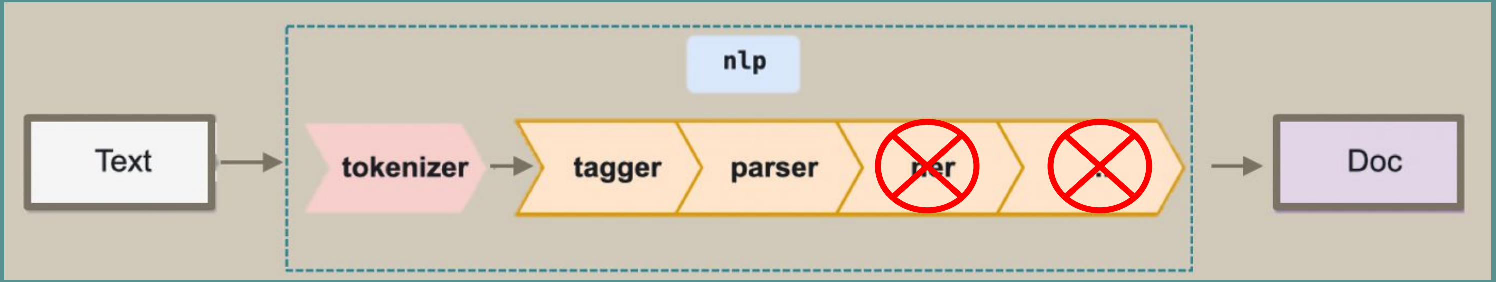
parser

ner

...

Doc

INPUT SENTENCE: 'The system as described above has its greatest application in an arrayed configuration of antenna elements .'



OUTPUT: [('configuration', 'Whole', 12), ('elements', 'Component', 15)]

Customizations - Tokenizer and Word Embedding

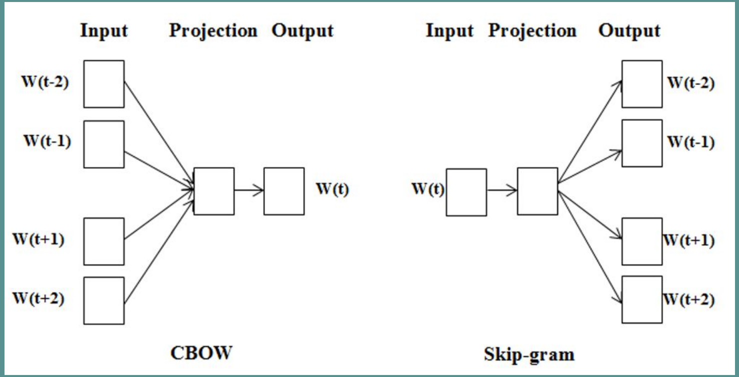
Original Sentence: The **glass-to-rubber** transition of poly **(BC-co-BS)** copolymers was investigated by DSC on melt quenched samples.

Default Tokenizer:

*(The, **glass, -, to, -, rubber**, transition, of, poly, **(, BC, -, co, -, BS,)**, copolymers, was, investigated, by, DSC, on, melt, quenched, samples, .)*

Custom Tokenizer:

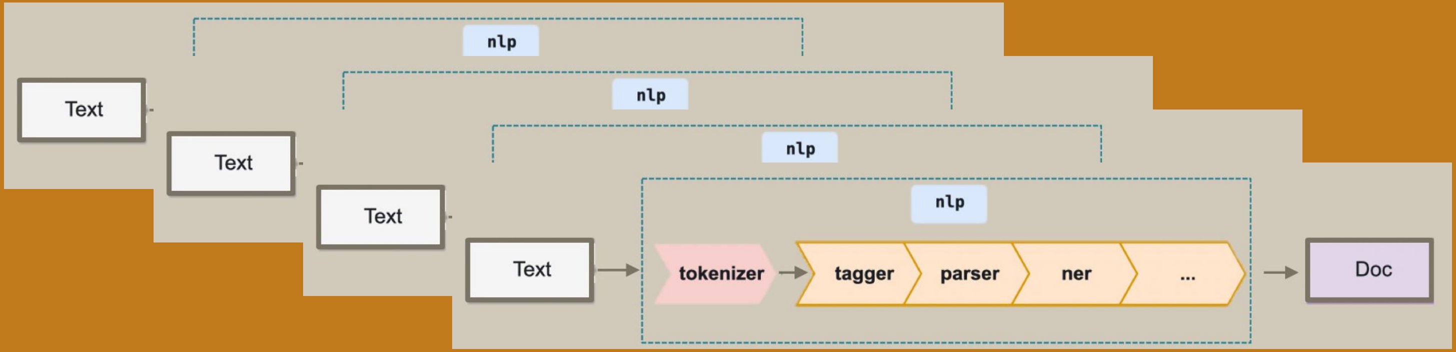
*(The, **glass-to-rubber**, transition, of, poly, **(, BC-co-BS,)**, copolymers, was, investigated, by, DSC, on, melt, quenched, samples, .)*



Scaling the pipeline

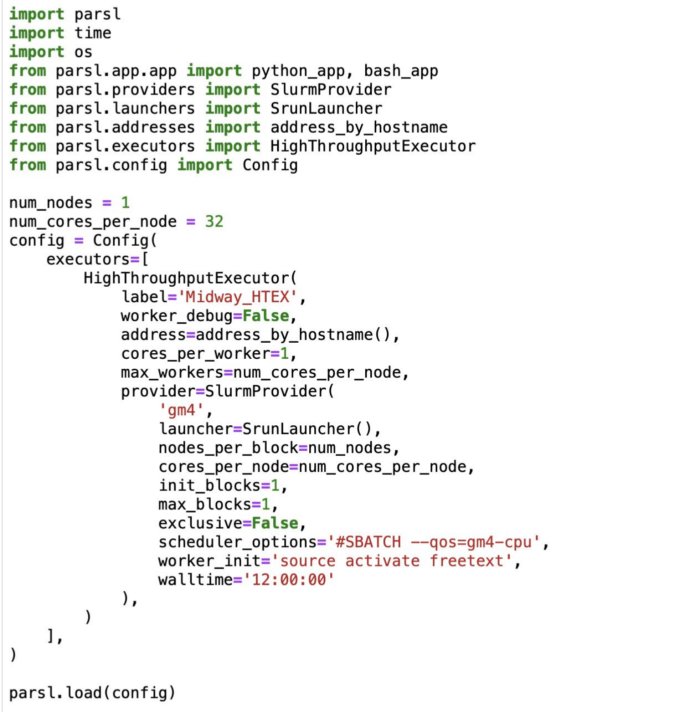
- Our models identify relations in fragments of text
- There are millions of papers to be processed

⇒ Scale by slicing the data and parallelizing across cores and nodes (with Parsl)

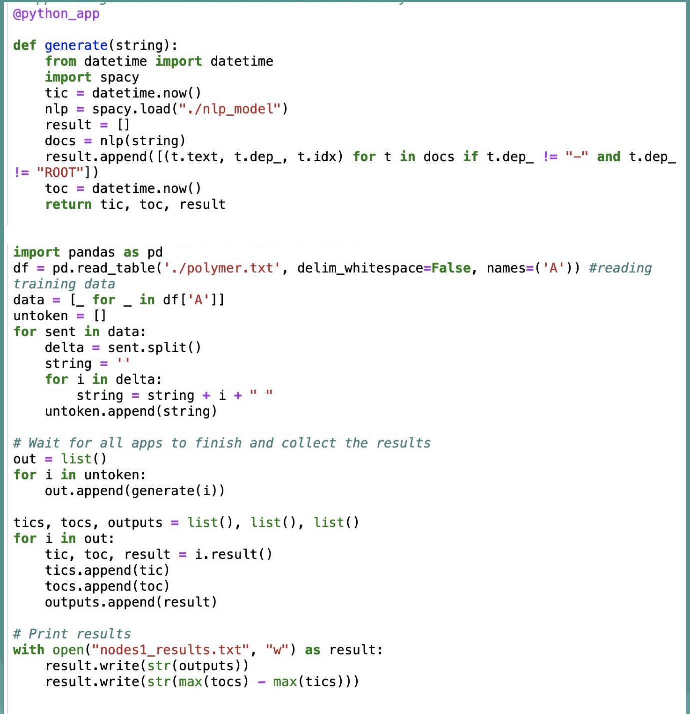


Scaling

- RCC Midway Cluster



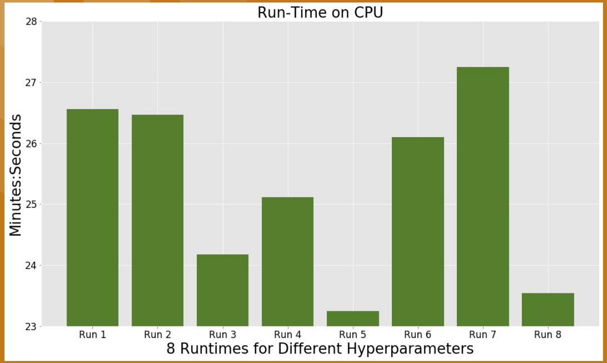
```python
import parsl
import time
import os
from parsl.app.app import python_app, bash_app
from parsl.providers import SlurmProvider
from parsl.launchers import SrunLauncher
from parsl.addresses import address_by_hostname
from parsl.executors import HighThroughputExecutor
from parsl.config import Config

num_nodes = 1
num_cores_per_node = 32
config = Config(
    executors=[
        HighThroughputExecutor(
            label='Midway_HTEX',
            worker_debug=False,
            address=address_by_hostname(),
            cores_per_worker=1,
            max_workers=num_cores_per_node,
            provider=SlurmProvider(
                'gm4',
                launcher=SrunLauncher(),
                nodes_per_block=num_nodes,
                cores_per_node=num_cores_per_node,
                init_blocks=1,
                max_blocks=1,
                exclusive=False,
                scheduler_options='#SBATCH --qos=gm4-cpu',
                worker_init='source activate freetext',
                walltime='12:00:00'
            ),
        )
    ],
)

parsl.load(config)
```

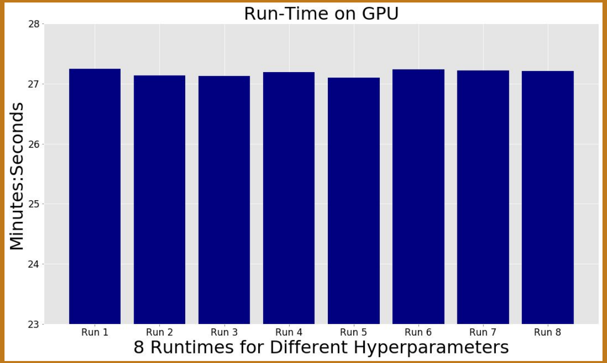


```python
@python_app

def generate(string):
    from datetime import datetime
    import spacy
    tic = datetime.now()
    nlp = spacy.load("./nlp_model")
    result = []
    docs = nlp(string)
    result.append([(t.text, t.dep_, t.idx) for t in docs if t.dep_ != "-" and t.dep_
!= "ROOT"])
    toc = datetime.now()
    return tic, toc, result


import pandas as pd
df = pd.read_table('./polymer.txt', delim_whitespace=False, names=('A')) #reading
training data
data = [_ for _ in df['A']]
untoken = []
for sent in data:
    delta = sent.split()
    string = ''
    for i in delta:
        string = string + i + " "
    untoken.append(string)

# Wait for all apps to finish and collect the results
out = list()
for i in untoken:
    out.append(generate(i))

tics, tocs, outputs = list(), list(), list()
for i in out:
    tic, toc, result = i.result()
    tics.append(tic)
    tocs.append(toc)
    outputs.append(result)

# Print results
with open("nodes1_results.txt", "w") as result:
    result.write(str(outputs))
    result.write(str(max(tocs) - max(tics)))
```

CPU vs GPU

- Used Model 3 and default CBOW parameters
- spaCy not optimized for GPU acceleration
- Fastest CPU runtime for training is 23 minutes

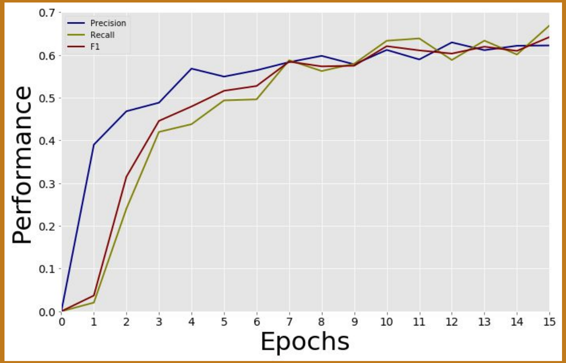




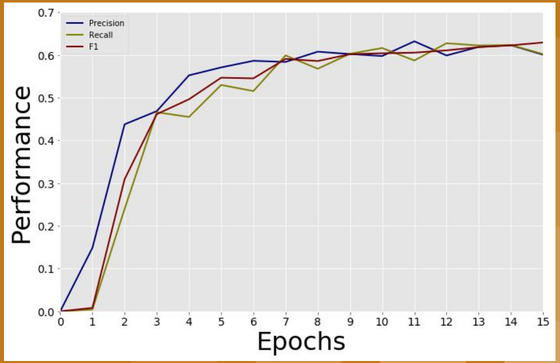
Performance

- Performance over 15 epochs for Skipgram and CBOW with default hyperparameters for Model 3

Performance Over 15 Epochs for CBOW

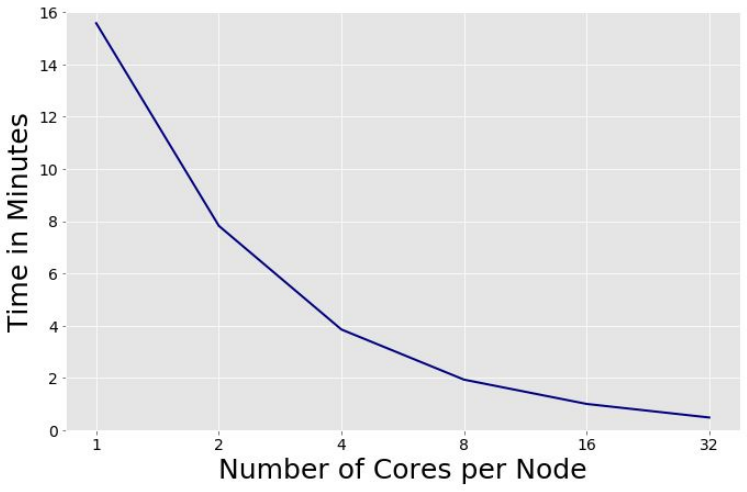


Performance Over 15 Epochs for Skip-gram

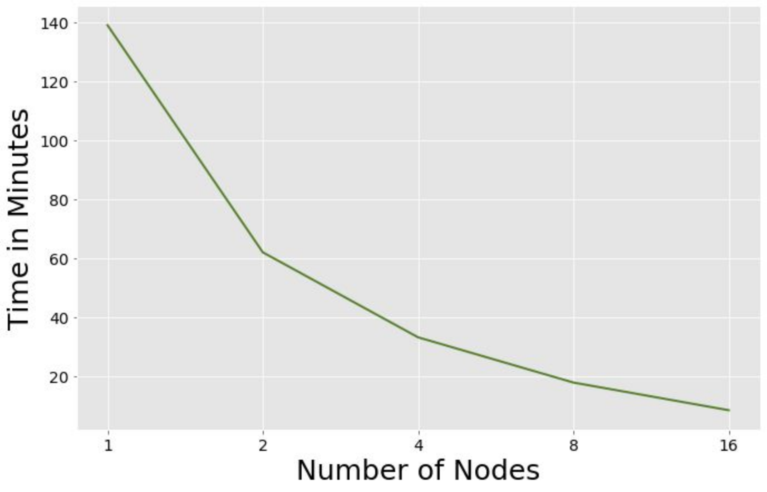


Scaling

Per Core Scaling



Per Node Scaling



Conclusion