Hands-on

Day 1: Comparing Representations

https://colab.research.google.com/drive/1JBW5pQ3-HxilyiJuZMREG8uBzzkQwuPS?usp=sharing

SemEval 2020 Task 1 Corpora

```
from languagechange.benchmark import SemEval2020Task1
```

```
semeval_en = SemEval2020Task1('EN')
```

```
corpus1 = semeval_en.corpus1_token
corpus2 = semeval_en.corpus2_token
```

```
corpus1 = semeval_en.corpus1_lemma
corpus2 = semeval_en.corpus2_lemma
```

Choose your corpus

```
import urllib.request

with open('brown_nolines.txt','w+') as g:
     url = 'http://www.sls.hawaii.edu/bley-vroman/brown_nolines.txt'
     for line in urllib.request.urlopen(url):
          g.write(line.decode('utf-8'))


from languagechange.corpora import LinebyLineCorpus

brown_corpus = LinebyLineCorpus('brown_nolines.txt', name='Brown Corpus', is_tokenized=False,
is_sentence_tokenized=True)
```

Static Embeddings

```
from languagechange.models.representation.staticimport StaticModel, CountModel, PPMI, SVD

# CORPUS 1
count_encoder1 = CountModel(corpus1, window_size=10, savepath='count_matrix1')
count_encoder1.encode()
PPMI_encoder1 = PPMI(count_encoder1, shifting_parameter=5, smoothing_parameter=0.75,
savepath='ppmi_matrix1')
PPMI_encoder1.encode()
svd_encoder1 = SVD(PPMI_encoder1, dimensionality=100, gamma=1.0, savepath='svd_count_matrix1')
svd_encoder1.encode()
svd_encoder1.load()
```

Extract Embeddings

```
encoder = svd_encoder1

def most_similar_idxs(word_idx, M, k):
    sims = np.dot(M[word_idx],M.T)
    return np.flip(np.argsort(sims))[:k]

M = np.asarray(encoder.matrix().todense())
M = preprocessing.normalize(M)
idx2word = encoder.row2word()
word2idx = {idx2word[i]:i for i in idx2word}

idx_word = word2idx['plane_nn']
idxs = most_similar_idxs(idx_word, M, 10)

vectors = M[idxs]
words = [idx2word[i] for i in idxs]
print(words)
```

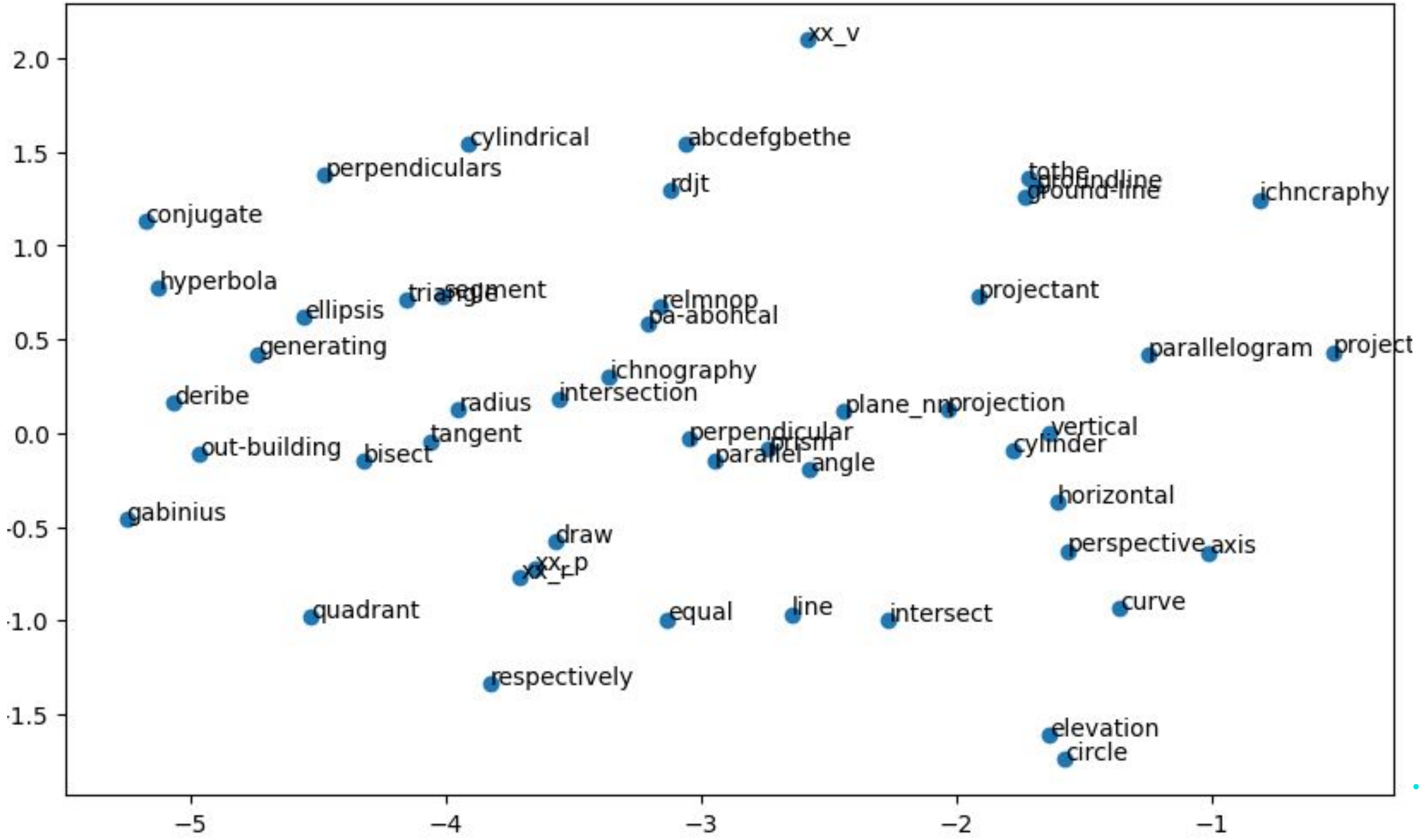
```
'plane_nn', 'projection', 'parallel',
'perpendicular', 'angle',
'ichnography', 'projectant',
'intersection', 'vertical', 'prism'
```

Extract Embeddings

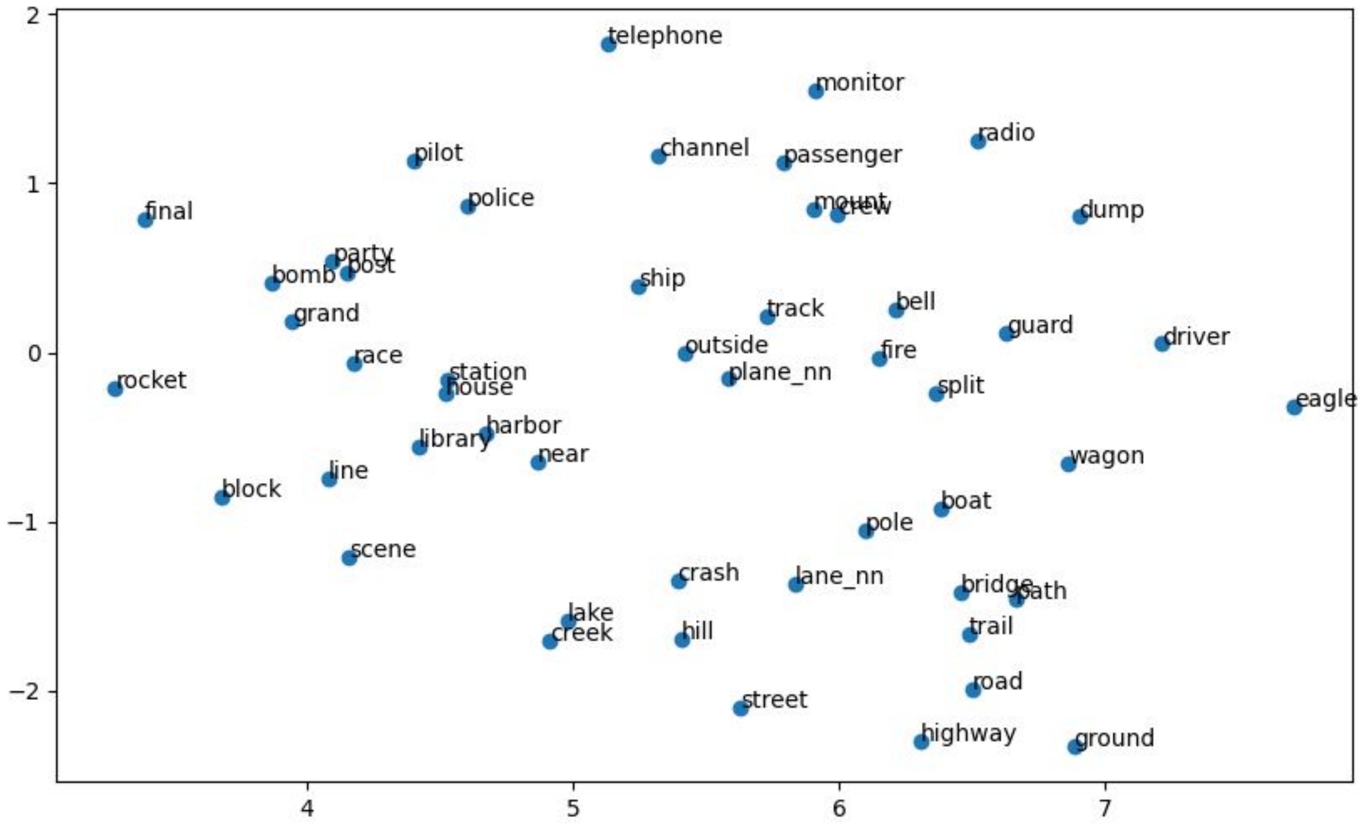
```
encoder = svd_encoder2

def most_similar_idxs(word_idx, M, k):
   sims = np.dot(M[word_idx],M.T)
   return np.flip(np.argsort(sims))[:k]

M = np.asarray(encoder.matrix().todense())
M = preprocessing.normalize(M)
idx2word = encoder.row2word()
word2idx = {idx2word[i]:i for i in idx2word}

idx_word = word2idx['plane_nn']
idxs = most_similar_idxs(idx_word, M, 10)

vectors = M[idxs]
words = [idx2word[i] for i in idxs]
print(words)
```

```
'plane_nn', 'ship', 'track', 'station',
'boat', 'outside', 'harbor', 'lane_nn',
'road', 'mount',
```

PPMI-SVD Embeddings (Corpus 1)



PPMI-SVD Embeddings (Corpus 2)



Align Static Embeddings

```
from sklearn.metrics.pairwise import cosine_similarity

plane1, plane2 = np.asarray(svd_encoder1['plane_nn']), np.asarray(svd_encoder2['plane_nn'])
will1, will2 = np.asarray(svd_encoder1['will']), np.asarray(svd_encoder2['will'])

print('plane1-plane2 similarity', cosine_similarity(plane1, plane2)[0][0])
print('will1-will2 similarity', cosine_similarity(will1, will2)[0][0])
```

```
plane1-plane2 similarity 0.17147865696062017
will1-will2 similarity 0.15674904444597207
```

Align Static Embeddings

```
from languagechange.models.representation.alignment import OrthogonalProcrustes

alignment = OrthogonalProcrustes('aligned1','aligned2')
alignment.align(svd_encoder1, svd_encoder2)

aligned1 = StaticModel('aligned1')
aligned2 = StaticModel('aligned2')
aligned1.load()
aligned2.load()
```

Align Static Embeddings

```
from sklearn.metrics.pairwise import cosine_similarity

plane1, plane2 = np.asarray(aligned1['plane_nn']), np.asarray(aligned2['plane_nn'])
will1, will2 = np.asarray(aligned1['will']), np.asarray(aligned2['will'])

print('plane1-plane2 similarity', cosine_similarity(plane1, plane2)[0][0])
print('will1-will2 similarity', cosine_similarity(will1, will2)[0][0])
```

```
plane1-plane2 similarity 0.20925707400117863
will1-will2 similarity 0.26262717664099955
```

Contextualized Embeddings

```python
from languagechange.models.representation.contextualized import BERT

target_words = list(semeval_en.binary_task.keys())
print([str(t) for t in target_words])

bert = BERT('bert-base-uncased', device='cuda')

usages = corpus1.search(target_words)

vectors = bert.encode(usages['graft_nn'])
print(vectors.shape)
```

XL-LEXEME

```python
from languagechange.models.representation.contextualized import XL_LEXEME

model = XL_LEXEME(device='cuda')

vectors = model.encode(usages['graft_nn'])
print(vectors.shape)
```

Exercise

Explore different words you think have changed (plane, graft, cell, gay, mouse).

Use different meaning representation models (count, PPMI, contextualized).

Report your findings qualitatively

Build your own visualization!

Challenge: Build your own visualization of Semantic Change

1. Choose a model: static embeddings, contextualized embeddings or XL-LEXEME
2. Choose a corpus
3. Choose a word that changed its meaning over time
4. Extract the embeddings
5. Plot the embeddings to show the semantic change

Challenge: Build your own visualization of Semantic Change

1. Choose a model: static embeddings, contextualized embeddings or XL-LEXEME
2. Choose a corpus
3. Choose a word that changed its meaning over time
4. Extract the embeddings
5. Plot the embeddings to show the semantic change

Challenge: Build your own visualization of Semantic Change

1. Choose a model: static embeddings, contextualized embeddings or XL-LEXEME
2. Choose a corpus
3. Choose a word that changed its meaning over time
4. Extract the embeddings
5. Plot the embeddings to show the semantic change

Challenge: Build your own visualization of Semantic Change

1. Choose a model: static embeddings, contextualized embeddings or XL-LEXEME
2. Choose a corpus
3. Choose a word that changed its meaning over time
4. Extract the embeddings
5. Plot the embeddings to show the semantic change

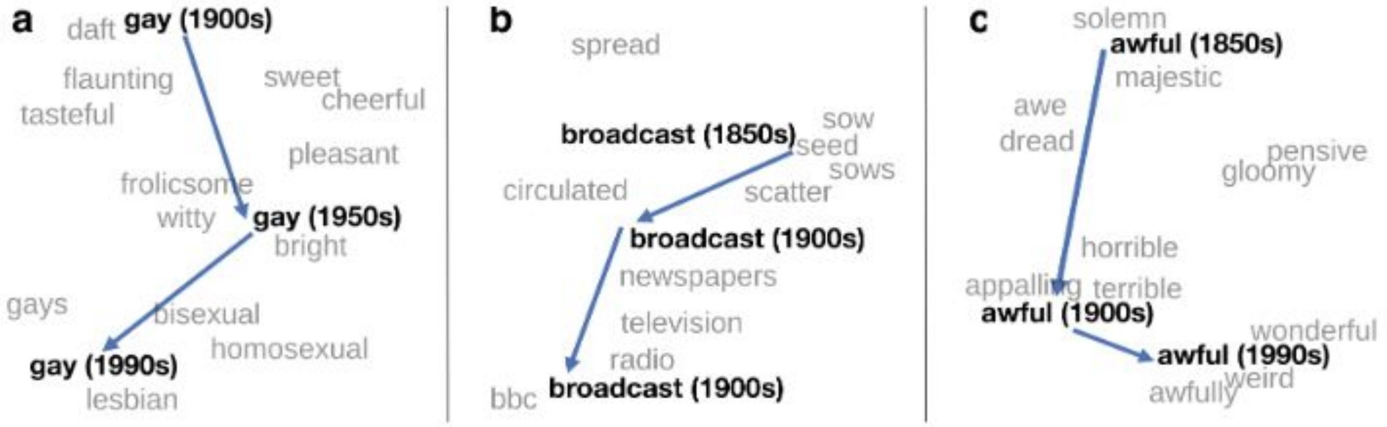
Challenge: Build your own visualization of Semantic Change

1. Choose a model: static embeddings, contextualized embeddings or XL-LEXEME
2. Choose a corpus
3. Choose a word that changed its meaning over time
4. Extract the embeddings
5. Plot the embeddings to show the semantic change

Challenge: Build your own visualization of Semantic Change

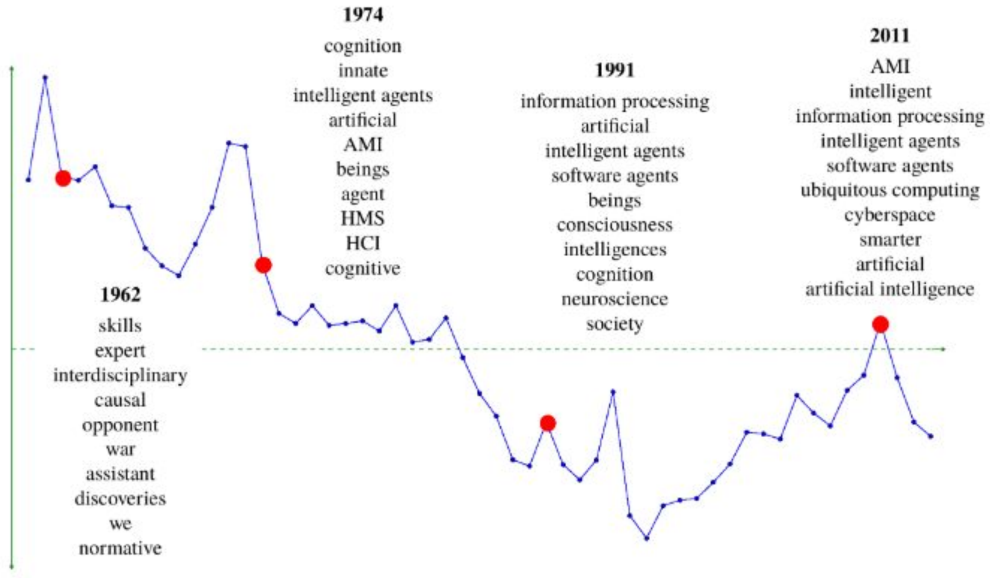
1. Choose a model: static embeddings, contextualized embeddings or XL-LEXEME
2. Choose a corpus
3. Choose a word that changed its meaning over time
4. Extract the embeddings
5. Plot the embeddings to show the semantic change

Previous visualizations (1)



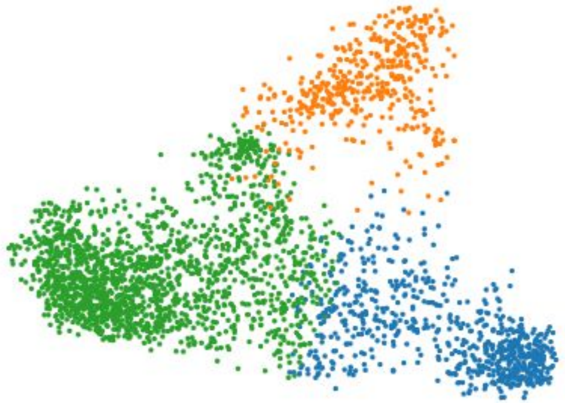
William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, Berlin, Germany. Association for Computational Linguistics.

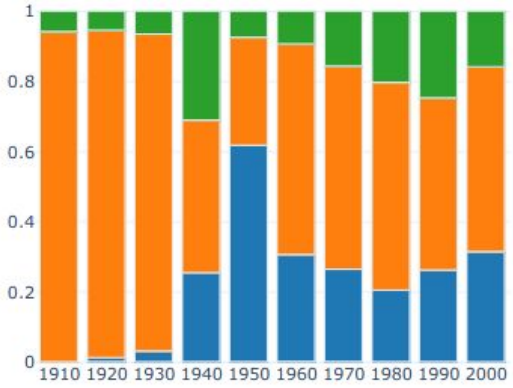
Previous visualizations (2)



Rudolph, Maja, and David Blei. "Dynamic embeddings for language evolution." In *Proceedings of the 2018 world wide web conference*, pp. 1003-1011. 2018.

Previous visualizations (3)


(a) PCA visualisation of the usage representations.



(b) Probability-based usage type distributions along time.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing Lexical Semantic Change with Contextualised Word Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

Show your visualization!