NLP For Economists

Lecture 3: NLP Methods and using off the shelf tools

Sowmya Vajjala

MGSE - LMU Munich
Guest Course, October 2022

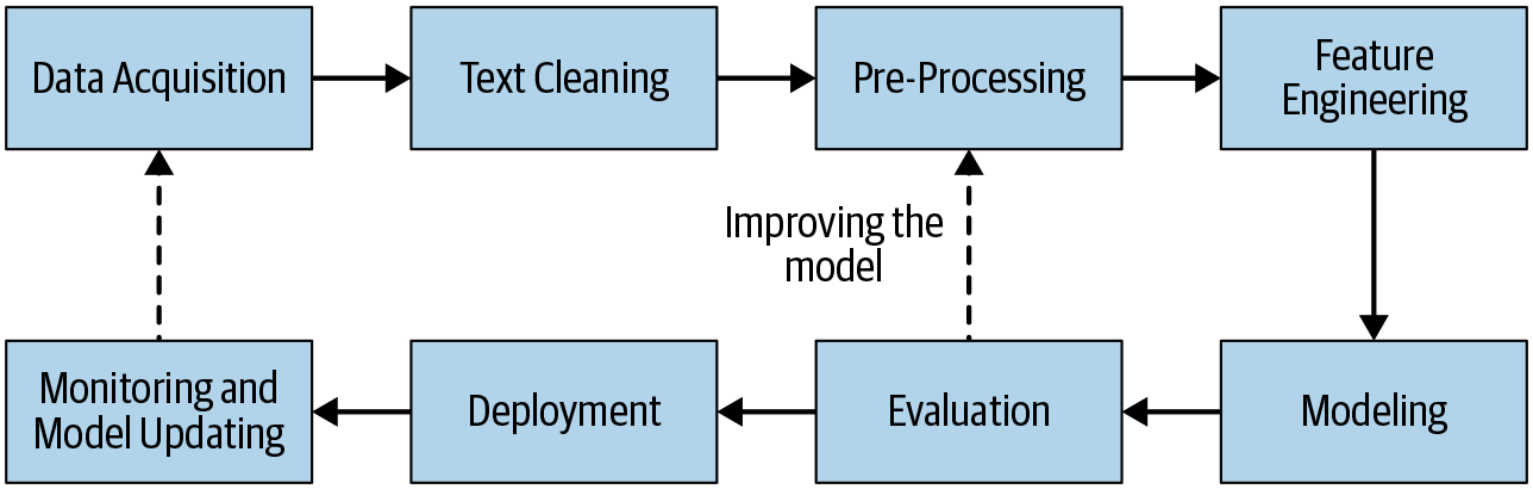
7th October 2022

Class Outline

- Some general overview of common NLP methods at various stages
- What could be missing for using NLP in economics?
- Using some existing python NLP libraries (Hands-on exercise)

NLP System Development Pipeline

A Common Case



My focus today

- I will focus on methods for data acquisition, feature engineering, and modeling.
- text extraction and cleaning, while important, is not traditionally treated as a part of NLP.
- pre-processing generally has a few standard steps, and is task dependent.
- evaluation: I will touch upon in the next class a bit.
- deployment and monitoring: they may not be relevant at this point, and methods are not NLP specific.

Data Collection Methods

Data Acquisition

Why do we need data at all?

my one line answer: to teach the machine!

Data Acquisition

Why do we need data at all?

my one line answer: to teach the machine!

- Modern NLP is heavily machine learning driven and machine learning approaches typically require lots and lots of examples to "train" on and learn a task.

Data Acquisition

Why do we need data at all?

my one line answer: to teach the machine!

- Modern NLP is heavily machine learning driven and machine learning approaches typically require lots and lots of examples to "train" on and learn a task.
- Assuming we are "engineering" everything manually, we still need some kind of curated data to evaluate our approach for its accuracy and coverage.

Data Acquisition

Why do we need data at all?

my one line answer: to teach the machine!

- Modern NLP is heavily machine learning driven and machine learning approaches typically require lots and lots of examples to "train" on and learn a task.
- Assuming we are "engineering" everything manually, we still need some kind of curated data to evaluate our approach for its accuracy and coverage.

So, good quality datasets are very (very) important for building any NLP system.

Why collect our own data?

- Clearly, building and evaluating NLP systems requires some corpus or the other.
- When we are working on developing new methods/algorithms, we often work with standard/pre-existing datasets to compare among different approaches.

Why collect our own data?

- Clearly, building and evaluating NLP systems requires some corpus or the other.
- When we are working on developing new methods/algorithms, we often work with standard/pre-existing datasets to compare among different approaches.
- However, when we are working on using NLP for a specific problem scenario, we won't often have such datasets that meet our exact needs.
- Data can come in various forms, but in most cases, we need some form of "labeled" data. (What's that?)

What kind of data do we need for NLP? -1

- Huge collections of text (language modeling, topic modeling etc)

What kind of data do we need for NLP? -1

- Huge collections of text (language modeling, topic modeling etc)
- Lot of examples of expected input $->$ expected output pairs. E.g.,
  - sentence-translated sentence pairs (machine translation)
  - spam/non-spam emails (spam classification)
  - question-answer pairs
  - sentence $->$ names of entities in it, relations between them etc (information extraction)

.... Data can come in various forms, but in most cases, we need some form of "labeled" data (i.e., input $->$ output pairs).

What kind of data do we need? - 2

- Quantity: Typically, "learning" methods are data hungry. The more, the better, although it may plateau at some point. (What is large?)

What kind of data do we need? - 2

- Quantity: Typically, "learning" methods are data hungry. The more, the better, although it may plateau at some point. (What is large?)
- Quality: Garbage in $->$ Garbage out. We can't take **anything** we can lay hands on. (Why?)

What kind of data do we need? - 2

- Quantity: Typically, "learning" methods are data hungry. The more, the better, although it may plateau at some point. (What is large?)
- Quality: Garbage in $->$ Garbage out. We can't take **anything** we can lay hands on. (Why?)
- Data without ethical concerns such as using data without consent, keeping personally identifiable information, racial/gender bias in training examples etc. (Why is this important?)

What kind of data do we need? - 2

- Quantity: Typically, "learning" methods are data hungry. The more, the better, although it may plateau at some point. (What is large?)
- Quality: Garbage in $->$ Garbage out. We can't take **anything** we can lay hands on. (Why?)
- Data without ethical concerns such as using data without consent, keeping personally identifiable information, racial/gender bias in training examples etc. (Why is this important?)
- (Ideally) Variety: spoken language, social media, literary texts, dialect variation, legal docs, non-native language, different topics/subjects etc (Why?)

How do we collect a corpus?

1. Use available data

There are many openly (most of them are free) accessible NLP datasets.

- Hugging Face NLP datasets
- Linguistic Data Consortium
- Researchers and organizations sometimes release their datasets for public (check research papers/company blog posts etc)

How do we collect a corpus?

2. Collect your own data

- Use existing data on the web
  - scraping websites (forums, newspapers, wikipedia etc)
  - collecting social media content (blog posts, tweets etc)
  - newspapers, wikipedia etc.
  - public archives (copyright free books, parliament proceedings, others)
- Collect your own (crowd sourcing, user studies, surveys etc)

Let us see some examples for each.

How do we collect a corpus?

2. Collect your own data

Scraping websites

- In a recently reported research work, researchers in USA collected a dataset of COVID-19 FAQs through webscraping (Process here)

How do we collect a corpus?

2. Collect your own data

Scraping websites

- In a recently reported research work, researchers in USA collected a dataset of COVID-19 FAQs through webscraping (Process here)
- This project describes how a government body in Mexico city used web scraping and NLP for improving job search experience.

How do we collect a corpus?

2. Collect your own data

Scraping websites

- In a recently reported research work, researchers in USA collected a dataset of COVID-19 FAQs through webscraping (Process here)
- This project describes how a government body in Mexico city used web scraping and NLP for improving job search experience.
- personal experiences: While working at SfS, I and Prof. Meurers collected data for training machine learning models using several websites (Weekly Reader, BBC Bitesize, Geo-Geolino (German), Time- Time for Kids etc)

How do we collect a corpus?

2. Collect your own data

Scraping websites

- In a recently reported research work, researchers in USA collected a dataset of COVID-19 FAQs through webscraping (Process here)
- This project describes how a government body in Mexico city used web scraping and NLP for improving job search experience.
- personal experiences: While working at SfS, I and Prof. Meurers collected data for training machine learning models using several websites (Weekly Reader, BBC Bitesize, Geo-Geolino (German), Time- Time for Kids etc)
- While working at a company long back, I collected question-answer pairs from a lot of forum websites, to build a search engine.

How do we collect a corpus?

2. Collect your own data

public archives, newspapers, Wikipedia etc

- Parliament proceedings from European Parliament (EUROPARL - several languages), Hansard in Canada (English-French), Nunavut Hansard (English-Inuktitut) etc. are commonly used as training corpora for machine translation.
- Crawled and annotated corpora from WSJ, NYT etc are frequently used as training data in NLP (you may have noticed if you browsed LDC).

How do we collect a corpus?

2. Collect your own data

public archives, newspapers, Wikipedia etc

- Parliament proceedings from European Parliament (EUROPARL - several languages), Hansard in Canada (English-French), Nunavut Hansard (English-Inuktitut) etc. are commonly used as training corpora for machine translation.
- Crawled and annotated corpora from WSJ, NYT etc are frequently used as training data in NLP (you may have noticed if you browsed LDC).

How do we collect a corpus?

2. Collect your own data

public archives, newspapers, Wikipedia etc

- Parliament proceedings from European Parliament (EUROPARL - several languages), Hansard in Canada (English-French), Nunavut Hansard (English-Inuktitut) etc. are commonly used as training corpora for machine translation.
- Crawled and annotated corpora from WSJ, NYT etc are frequently used as training data in NLP (you may have noticed if you browsed LDC).
- personal experiences: While working at a company, I scraped Canadian supreme court websites for case summaries.

How do we collect a corpus?

2. Collect your own data

public archives, newspapers, Wikipedia etc

- Parliament proceedings from European Parliament (EUROPARL - several languages), Hansard in Canada (English-French), Nunavut Hansard (English-Inuktitut) etc. are commonly used as training corpora for machine translation.
- Crawled and annotated corpora from WSJ, NYT etc are frequently used as training data in NLP (you may have noticed if you browsed LDC).
- personal experiences: While working at a company, I scraped Canadian supreme court websites for case summaries.
- I also crawled Wikipedia to build a labeled dataset of various categories of articles related to legal domain (e.g., finance law, human rights law etc) to develop a document tagger.

How do we collect a corpus?

2. Collect your own data

Social media content

- Twitter is a common source of data to study various issues such as trending topics, fake news, offensive text detection, sentiment analysis etc.
- It is also useful in industry scenarios for customer support, product reviews etc.
- Other social media websites such as Facebook, Reddit etc are also regularly used to collect data in NLP.
- This link shows how common it is to use such data for NLP Research!

How do we collect a corpus?

2. Collect your own data

crowd sourcing, user studies etc

- Crowdsourced datasets from https://msropendata.com/datasets?term=crowdsourcing and other such large organizations (Google etc)

How do we collect a corpus?

2. Collect your own data

crowd sourcing, user studies etc

- Crowdsourced datasets from https://msropendata.com/datasets?term=crowdsourcing and other such large organizations (Google etc)
- Crowdsourcing for compamy's internal use: Google collects translation data through user contributions

How do we collect a corpus?

2. Collect your own data

crowd sourcing, user studies etc

- Crowdsourced datasets from https://msropendata.com/datasets?term=crowdsourcing and other such large organizations (Google etc)
- Crowdsourcing for compamy's internal use: Google collects translation data through user contributions
- personal experiences: In 2016, Prof Meurers and I collected eye-tracking experiment data to study an NLP problem, by collaborating with cognitve science researchers at IWM-Tuebingen.
- in 2018, I and my student collected data through a user study where university students read texts and answered questions about them.

How do we collect a corpus?

2. Collect your own data

crowd sourcing, user studies etc

- Crowdsourced datasets from https://msropendata.com/datasets?term=crowdsourcing and other such large organizations (Google etc)
- Crowdsourcing for compamy's internal use: Google collects translation data through user contributions
- personal experiences: In 2016, Prof Meurers and I collected eye-tracking experiment data to study an NLP problem, by collaborating with cognitve science researchers at IWM-Tuebingen.
- in 2018, I and my student collected data through a user study where university students read texts and answered questions about them.
- Our own data can also come from internal organizational data sources like search logs, customer support data etc.

How do we collect a corpus?

"Annotated" Data?

- In some of these examples (scraping q&a, crawling Wikipedia with categories), we see data that does not need additional annotations.

How do we collect a corpus?

"Annotated" Data?

- In some of these examples (scraping q&a, crawling Wikipedia with categories), we see data that does not need additional annotations.
- In others, we wonder where the labels (e.g., positive vs negative sentiments, fake news vs normal), where does the labeled/annotated data come from?

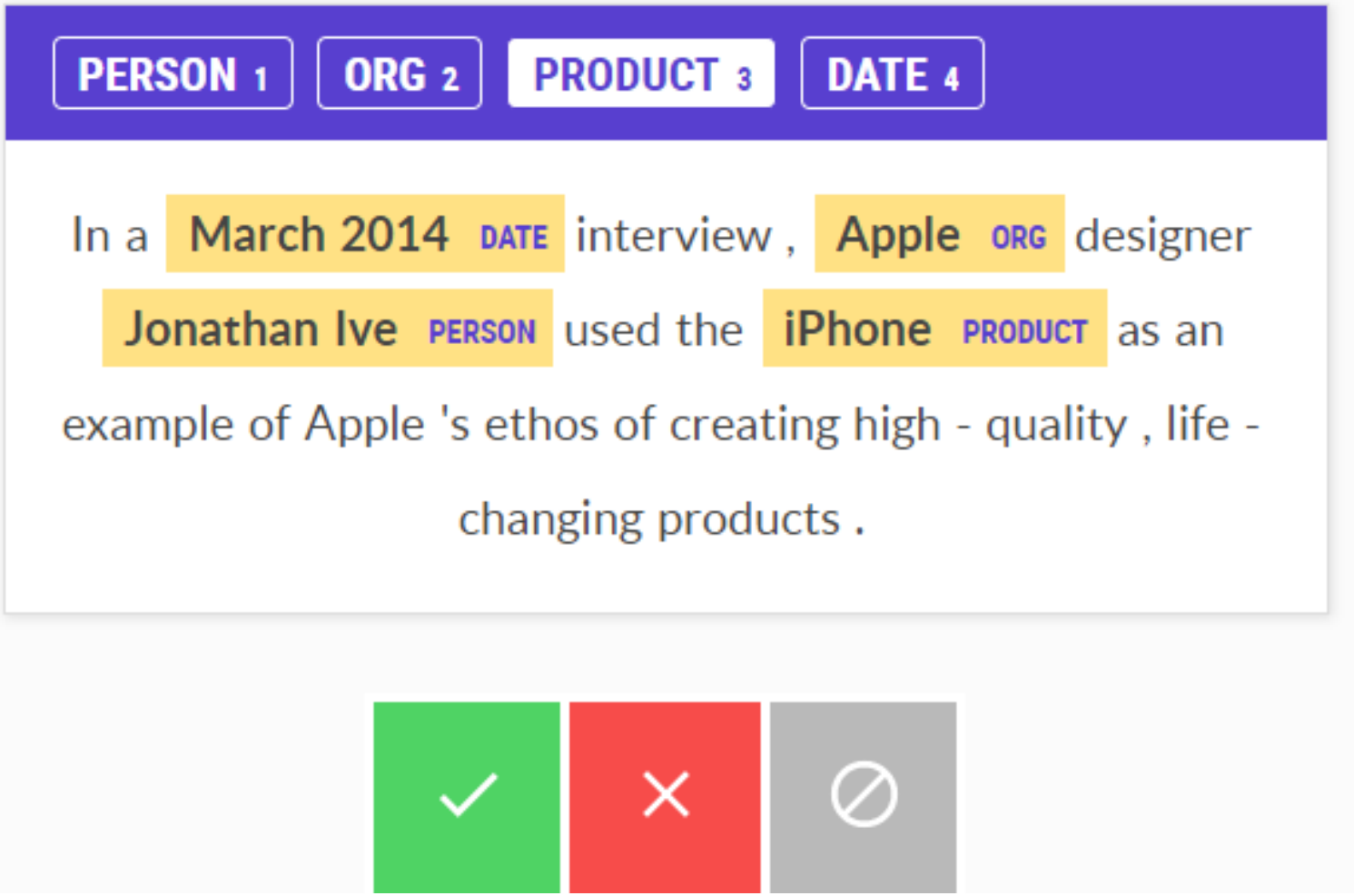
How do we collect a corpus?

"Annotated" Data?

- In some of these examples (scraping q&a, crawling Wikipedia with categories), we see data that does not need additional annotations.
- In others, we wonder where the labels (e.g., positive vs negative sentiments, fake news vs normal), where does the labeled/annotated data come from?
- crowd sourcing or having a more structured annotation experiments are two common ways of getting them.

How do we collect a corpus?

Active Learning



Source

How do we collect a corpus?

3. "Generate" your own data

- Data labeling: Look for patterns in the data, and generate labels using string matching, regular expressions etc.
- Data Augmentation: Generate synthetic data to add to what is already there through various strategies like - replacing words with synonyms, back translation, replacing names etc.
- Active Learning: interactively labeling the text to teach the learning algorithm

(More on some of these in Session 5)

Some practical issues with data collection

- Clearly, there are benefits with scraping data from websites of various kinds.
- What are some potential risks?

Some practical issues with data collection

- Clearly, there are benefits with scraping data from websites of various kinds.
- What are some potential risks?
  1. user created, publicly available data can pose privacy risks for individuals who created it.

Some practical issues with data collection

- Clearly, there are benefits with scraping data from websites of various kinds.
- What are some potential risks?
  1. user created, publicly available data can pose privacy risks for individuals who created it. (sometimes, datasets are released after anonymization).
  2. you may be collecting the data without asking for permissions (ask: is everything freely visible on the web essentially free for such use?)
  3. copyrights/terms of service conflicts may come up.

Some practical issues with data collection

- Clearly, there are benefits with scraping data from websites of various kinds.
- What are some potential risks?
  1. user created, publicly available data can pose privacy risks for individuals who created it. (sometimes, datasets are released after anonymization).
  2. you may be collecting the data without asking for permissions (ask: is everything freely visible on the web essentially free for such use?)
  3. copyrights/terms of service conflicts may come up.
  4. Often, websites' policies allow free browsing, but not crawling.

Diesner, J., & Chin, C. L. (2016, May). Gratis, libre, or something else? Regulations and misassumptions related to working with publicly available text data. In Actes du Workshop on Ethics In Corpus Collection, Annotation & Application (ETHI-CA2), LREC, Portoroz, Slovénie.

Datasheets for Datasets-1

Ask the following while creating/using a dataset

1. Why was the dataset created? Who funded its creation? How and when was it created?
2. Who were involved in the collection? (students, crowdworkers etc.) How were they compensated?
3. What preprocessing/cleaning was done?
4. How is the dataset released/distributed?
5. Will the dataset be updated? How often? by whom?

Source: "Datasheets for Datasets", Gebru et.al., 2018

Datasheets for Datasets-2

Legal and Ethical considerations

- If it relates to people, were they told what the dataset would be used for and did they consent?
- If it relates to people, could this dataset expose people to harm or legal action?
- If it relates to people, does it unfairly advantage or dis-advantage a particular social group?
- Does the dataset comply with the EU General Data Protection Regulation (GDPR)?

Full list of questions in the original paper linked in the previous slide. Bender & Friedman, 2018 is another good paper on the topic of corpora creation/use .

A small exercise

Visit: https://huggingface.co/datasets. You can search some of the openly accessible NLP datasets there. Pick one or two datasets, and check their descriptions to see what information they provide about how they collected the data, what information is missing etc.
Time: 10 minutes

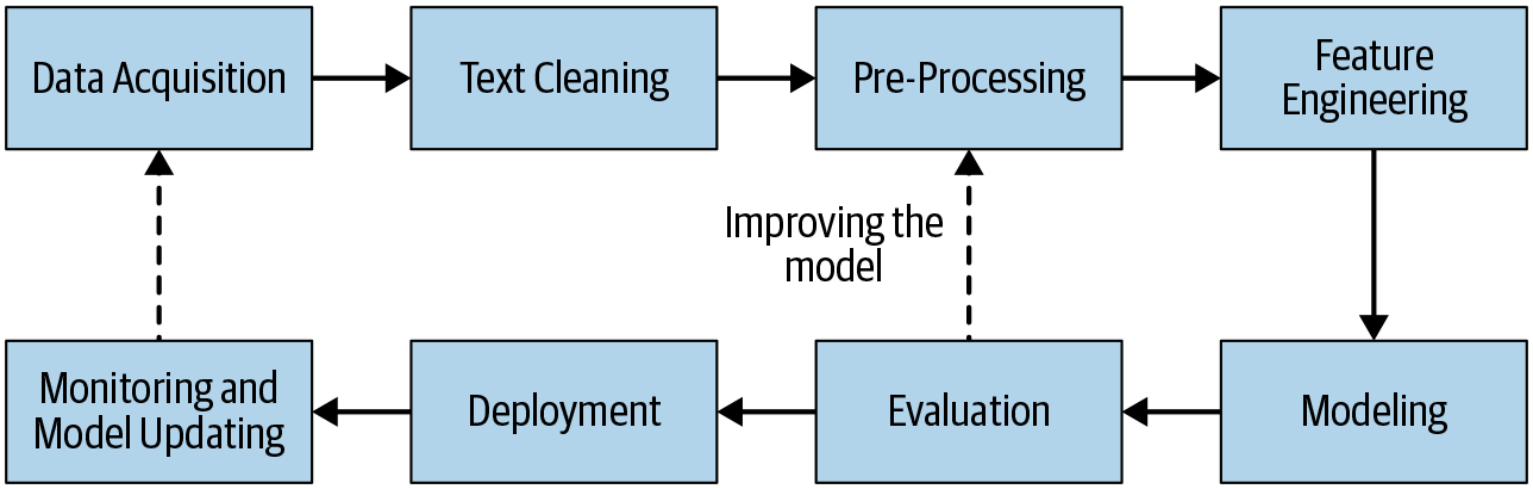
Share your observations

Data Acquisition: Summary

- There are many ways to get data, and label it.
- There are some concerns to be addressed while doing it too.
- I hope this part of today's class gave you a broad overview of all things related to data in NLP.

NLP System Development Pipeline

A Common Case



Text Extraction

- Extracting texts from different file formats
- Traditionally not a part of NLP but important for NLP nevertheless.
- An important thing to note is: it can be very hard when you are dealing with non-plain text formats such as pdfs, images etc.

Text Pre-processing

- Sentence segmentation and word tokenization

Text Pre-processing

- Sentence segmentation and word tokenization
- Stop word removal, stemming and lemmatization, removing digits/punctuation, lowercasing, etc.

Text Pre-processing

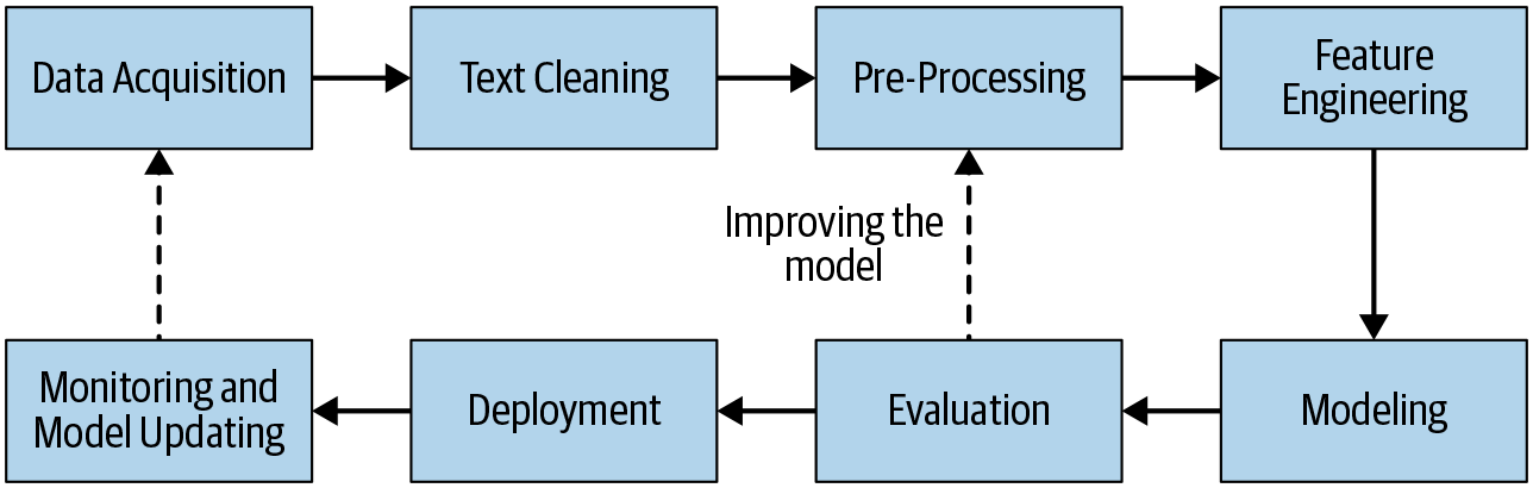
- Sentence segmentation and word tokenization
- Stop word removal, stemming and lemmatization, removing digits/punctuation, lowercasing, etc.
- Normalization, language detection, code mixing, transliteration, etc.

Text Pre-processing

- Sentence segmentation and word tokenization
- Stop word removal, stemming and lemmatization, removing digits/punctuation, lowercasing, etc.
- Normalization, language detection, code mixing, transliteration, etc.
- POS tagging, parsing, coreference resolution, etc.

- what pre-processing you do is very task and process dependent!

NLP System Development Pipeline



Feature Extraction

- Feature extraction is about representing the text in some form of numeric vector.
- It can be a hand-crafted, dense representation where each item in the vector has specific information encoded in it.
  $\rightarrow$ let us say I have two features: (number of words, number of sentences), I can represent any text as a two dimensional vector.
- It can also be automatic (e.g., bag of words, text embeddings etc).

Question: How do you combine both? Can you?

Evaluation

- Evaluation measures are task specific.
- Common measures seen across many tasks: accuracy, precision, recall, F1 score etc.
- Task specific ones: BLEU (Machine Translation), ROUGE (Text summarization) etc.
- evaluate is a python library that implements several NLP evaluation measures across tasks.
  - You can also try these online on huggingface's interactive widget

Rest of the steps aren't specific to NLP. So, skipping them.

Questions so far?

Modeling: Rule based approaches

- Idea: Know the problem, code some heuristics tailor made to the tasks.
- Tools used: word lists, regular expressions etc
- Early approaches were rule-based. Current approaches are dominated by Deep Learning but the rules have their place even now and it is important to be familiar with the approach.

Rule based approach: An Example

Recognizing negation in text-1

Start with a list of negative words

```
NEGATE =  ["aint", "arent", "cannot", "cant", "couldnt", "darent", "didnt", "doesnt",
  "ain't", "aren't", "can't", "couldn't", "daren't", "didn't", "doesn't",
  "dont", "hadnt", "hasnt", "havent", "isnt", "mightnt", "mustnt", "neither",
  "don't", "hadn't", "hasn't", "haven't", "isn't", "mightn't", "mustn't",
  "needn't", "needn't", "never", "none", "nope", "nor", "not", "nothing", "nowhere",
  "oughtnt", "shant", "shouldnt", "uhuh", "wasnt", "werent",
  "oughtn't", "shan't", "shouldn't", "uh-uh", "wasn't", "weren't",
  "without", "wont", "wouldnt", "won't", "wouldn't", "rarely", "seldom", "despite"]
```

source: VaderSentiment, a rule based sentiment analyzer (https://github.com/cjhutto/vaderSentiment)

Rule based approach: An Example

Recognizing negation in text-2

Use the list to understand if a text has negation

```
def negated(input_words, include_nt=True):
"""
Determine if input contains negation words
"""
input_words = [str(w).lower() for w in input_words]
neg_words = []
neg_words.extend(NEGATE)
for word in neg_words:
    if word in input_words:
        return True
if include_nt:
    for word in input_words:
        if "n't" in word:
            return True
'''if "least" in input_words:
    i = input_words.index("least")
    if i > 0 and input_words[i - 1] != "at":
        return True'''
return False
```

Comments

- You don't need any data for a rule based approach (you will need some, to evaluate, of course)
- But you would need a strong domain knowledge to come up with the heuristics
- Coverage of the rules for the language phenomenon one may encounter in that domain is not guaranteed.

General framework for NLP systems

- In general, NLP systems follow a "supervised" approach.
- We collect a corpus of X $->$ Y examples and start with "building" an NLP system.
- We build a few models and compare across them on some test data, to choose one final model.
- In current day NLP, it is common to combine feature engineering and modeling steps together, due to the use of deep learning methods to represent text.

Building NLP Models: Data Splits

1. Train set: a larger subset of the data, used for training models
2. Development/Validation set: Smaller set used for choosing the best model/parameters among the different ones explored.
3. Test set: Used to do a final evaluation of the NLP model

Modeling: Features + Machine Learning

- Step 1: Represent texts as a feature vector (We saw an example of bag of words vectors in the first class).
- Use the (vector, label) pairs as input to any standard machine learning approach (e.g., logistic regression, support vector machine, decision trees etc - I will talk briefly about these in the next class)
- Evaluate and compare among approaches.

Modeling: Features + Machine Learning

Handcrafted features

```
def extract_features(X):
    features = {}
    X_split = X.split(' ')
    # Count the number of "good" and "bad" words
    good_words = ['great', 'good']
    bad_words = ['worst', 'bad']
    for x in X_split:
        if x in good_words:
            features['pos'] = features.get('pos', 0) + 1
        if x in bad_words:
            features['neg'] = features.get('neg', 0) + 1
    return features

print(extract_features("Is this a good test or a bad one?"))
```

The feature representation for the text will be:
{'pos': 1, 'neg': 1}.

this can be a two-dimensional vector [1,1] in practice.

source

Comments

- Domain knowledge has less role to play, and it is possible to automate the feature generation (using vectorization methods like bag of words etc)
- We would need some amount of data so that the machine learning algorithm can "learn" the patterns instead of us explicitly coding them.
- The hope is that lots of data will free us from coverage issues of hand crafted rules, and the machine is capable of learning patterns from large amounts of data anyway.

Modeling: End-to-End systems with Deep Learning

Contemporary NLP is dominated by this approach.

- We rely on large, pre-trained language models that are freely available, as a starting point.
- We "fine-tune" them on our data
- The fine-tuning process is pretty much the same whatever the task is but format of the input and output differs based on the task

Comments

- Combines feature engineering and modeling, reducing human intervention further.
- Recent approaches such as "transfer learning" and "few shot", "zero shot" learning are even reducing the dependence on large amounts of data.

Let us take a short break here. 5 minutes.

NLP with Deep Learning: An overview

Text Embeddings: The backbone of contemporary NLP

- All end to end approaches rely on some form of neural network based vectorization of text, which are called "embeddings".
- Embeddings rely on the intuition that words that occur in similar contexts may be related in some way.
- This is known as the "distributional hypothesis"

Distributional Hypothesis

- Words that occur in similar contexts may be related in some way (synonyms, antonyms etc)
- The link between similarity in how words are distributed and similarity in what they mean is called the distributional hypothesis.
- This hypothesis was studied closely since the 1950s and it extended to the concept of "vector semantics" which forms the basis of neural text representations known as embeddings in NLP.

Slides that follow are based on Chapter 6 in Jurafsky & Martin, 3rd Edition

Words and their relationships

If we then take every occurrence of each word (say **strawberry**) and count the context words around it, we get a word-word co-occurrence matrix. Fig. 6.6 shows a simplified subset of the word-word co-occurrence matrix for these four words computed from the Wikipedia corpus (Davies, 2015).

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Note in Fig. 6.6 that the two words *cherry* and *strawberry* are more similar to each other (both *pie* and *sugar* tend to occur in their window) than they are to other words like *digital*; conversely, *digital* and *information* are more similar to each other than, say, to *strawberry*. Fig. 6.7 shows a spatial visualization.

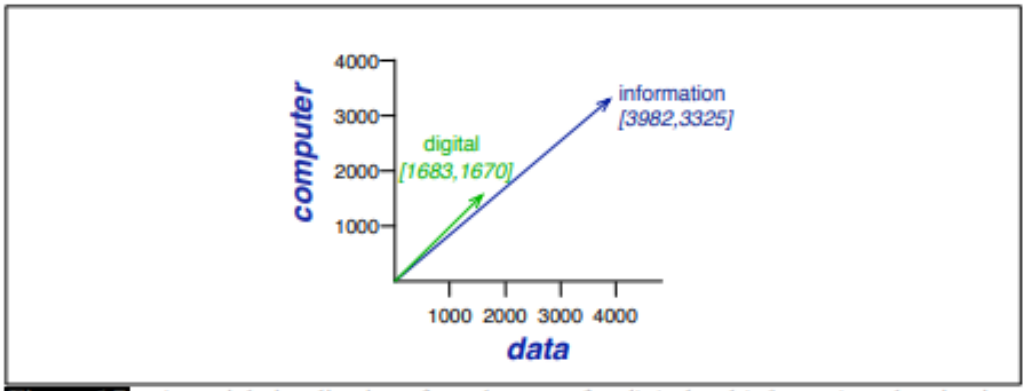


Figure 6.7 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

source: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Similarity between two words

The cosine value ranges from 1 for vectors pointing in the same direction, through 0 for orthogonal vectors, to -1 for vectors pointing in opposite directions. But since raw frequency values are non-negative, the cosine for these vectors ranges from 0–1.

Let's see how the cosine computes which of the words *cherry* or *digital* is closer in meaning to *information*, just using raw counts from the following shortened table:

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry, information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

$$\cos(\text{digital, information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

The model decides that *information* is way closer to *digital* than it is to *cherry*, a result that seems sensible. Fig. 6.8 shows a visualization.

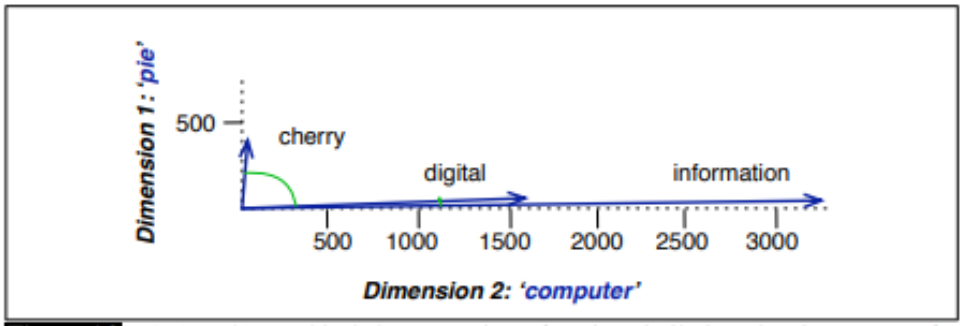


Figure 6.8 A (rough) graphical demonstration of cosine similarity, showing vectors for three words (*cherry*, *digital*, and *information*) in the two dimensional space defined by counts of the words *computer* and *pie* nearby. Note that the angle between *digital* and *information* is smaller than the angle between *cherry* and *information*. When two vectors are more similar, the cosine is larger but the angle is smaller; the cosine has its maximum (1) when the angle between two vectors is smallest (0°); the cosine of all other angles is less than 1.

source: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Intuition

For example, suppose you didn't know the meaning of the word *ongchoi* (a recent borrowing from Cantonese) but you see it in the following contexts:

(6.1) Ongchoi is delicious sauteed with garlic.

(6.2) Ongchoi is superb over rice.

(6.3) ...ongchoi leaves with salty sauces...

And suppose that you had seen many of these context words in other contexts:

(6.4) ...spinach sauteed with garlic over rice...

(6.5) ...chard stems and leaves are delicious...

(6.6) ...collard greens and other salty leafy greens

The fact that *ongchoi* occurs with words like *rice* and *garlic* and *delicious* and *salty*, as do words like *spinach*, *chard*, and *collard greens* might suggest that ongchoi is a leafy green similar to these other leafy greens.[1] We can do the same thing computationally by just counting words in the context of *ongchoi*.

source: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Vector Semantics

- Vector semantics is the standard way to represent word meaning in NLP.
- Vector semantic approaches "learn" to define the meaning of a word by its distribution in language use i.e., its neighboring words or grammatical environment.

Vector Semantics

- Vector semantics is the standard way to represent word meaning in NLP.
- Vector semantic approaches "learn" to define the meaning of a word by its distribution in language use i.e., its neighboring words or grammatical environment.
- The goal of such a model is to represent a word as a point in a multidimensional semantic space (called an "embedding") that is derived from distributions of its neighbors.
- The question of how to "learn" this multidimensional space resulted in various approaches to word embeddings in NLP.

What do Embeddings learn?

Figure 6.1 A two-dimensional (t-SNE) projection of embeddings for some words and phrases, showing that words with similar meanings are nearby in space. The original 60-dimensional embeddings were trained for sentiment analysis. Simplified from Li et al. (2015) with colors added for explanation.

source: https://web.stanford.edu/~jurafsky/slp3/6.pdf

Earlier text representations vs embeddings

- Count based vs learnt representations
- Sparse vectors with mostly zeros vs Dense vectors with non-zero values in all dimensions.
- Long vectors, scaling with vocabulary size vs Short vectors with limited dimensions.

Why bother about embeddings?

- Because embeddings are everywhere now!
- It turns out that dense vectors typically work better in every NLP task than sparse vectors.

Why bother about embeddings?

- Because embeddings are everywhere now!
- It turns out that dense vectors typically work better in every NLP task than sparse vectors.
- Although we don't fully know why yet, it is possible that:
  - Shorter and denser vectors will need fewer parameters to learn better for ML/DL models.
  - Dense vectors may be capturing synonymy and other relations between words.

How do we use these?

- Pre-trained embeddings are available for various kinds of text.
- If you want to train your own word2vec embeddings for a specific problem domain, it is easy and fast.
- Instead of using bag of words/ngrams etc, you can just use them as features.
- How?: A common strategy is to get embeddings for individual words from this model, and average them to get full text feature representation

Using pre-trained word2vec

```
from gensim.models import Word2Vec, KeyedVectors
pretrainedpath = '/tmp/input/GoogleNews-vectors-negative300.bin.gz'
w2v_model = KeyedVectors.load_word2vec_format(pretrainedpath, binary=True)
#loads the model

#What is the vector representation for a word?
w2v_model['computer']
```

source

What do Embeddings learn?

```
In [6]: #What is the vector representation for a word?
        w2v_model['computer']
```

```
Out[6]: array([ 1.07421875e-01, -2.81171875e-01,  1.23046875e-01,  2.11914062e-01,
               -9.13085938e-02,  2.16796875e-01, -1.31835938e-01,  8.30078125e-02,
                2.82148438e-01,  4.78515625e-02,  3.66210938e-02, -2.45361328e-02,
                2.39257812e-02, -1.68156250e-01, -2.61230469e-02,  9.71679688e-02,
               -6.34765625e-02,  1.84570312e-01,  1.70898438e-01, -1.63085938e-01,
               -1.09375000e-01,  1.49414062e-01, -4.65393066e-04,  9.61914062e-02,
                1.68945312e-01,  2.68925293e-03,  8.93554688e-02,  6.49414062e-02,
                3.56445312e-02, -6.93359375e-02, -1.46484375e-01, -1.21093750e-01,
               -2.27539062e-01,  2.45361328e-02, -1.24511719e-01, -3.18359375e-01,
               -2.20703125e-01,  1.38859375e-01,  3.66210938e-02, -3.63769531e-02,
               -1.13281250e-01,  1.95312500e-01,  9.76562500e-02,  1.26953125e-01,
                6.59179688e-02,  6.93359375e-02,  1.02539062e-02,  1.75781250e-01,
               -1.68945312e-01,  1.21307373e-03, -2.98828125e-01, -1.15234375e-01,
                5.66406250e-02, -1.77734375e-01, -2.88984375e-01,  1.76757812e-01,
                2.38837109e-02, -2.57812500e-01, -4.46777344e-02,  1.88476562e-01,
                5.51757812e-02,  5.82929688e-02, -1.06933594e-01,  1.89453125e-01,
               -1.16210938e-01,  8.49609375e-02, -1.71875000e-01,  2.45117188e-01,
               -1.73828125e-01, -8.30078125e-03,  4.56542969e-02, -1.61132812e-02,
                1.86523438e-01, -6.85468750e-02, -4.17480469e-02,  1.82617188e-01,
                2.20703125e-01, -1.22558594e-01, -2.55126953e-02, -3.88593750e-01,
                9.13085938e-02,  1.68156250e-01,  1.70898438e-01,  1.19628906e-01,
                7.80078125e-02, -2.64892578e-02, -3.08837891e-02,  4.86250000e-01,
               -1.01562500e-01,  5.71289062e-02, -7.26318359e-03, -9.17968750e-02,
               -1.50390625e-01, -2.55859375e-01,  2.16796875e-01, -3.63769531e-02,
                2.24609375e-01,  8.88781250e-02,  1.56250000e-01,  5.27343750e-02,
                1.50390625e-01, -1.14746094e-01, -8.64257812e-02,  1.19140625e-01,
               -7.17773438e-02,  2.73437500e-01, -1.64062500e-01,  7.29378117e-03,
                4.21875000e-01, -1.12792969e-01, -1.35742188e-01, -1.31835938e-01,
               -1.37695312e-01, -7.66601562e-02,  6.25000000e-02,  4.98046875e-02,
               -1.91406250e-01, -6.83027344e-02,  2.27539062e-01,  5.88378906e-02,
               -3.24218750e-01,  5.41992188e-02, -1.35742188e-01,  8.17871094e-03,
               -5.24902344e-02, -1.74713135e-03, -9.81445312e-02, -2.86865234e-02,
                3.61328125e-02,  2.15820312e-01,  5.98144531e-02, -3.88593750e-01,
               -2.27539062e-01,  2.61718750e-01,  9.86328125e-02, -5.07812500e-02,
                1.78222656e-02,  1.31835938e-01, -5.35156250e-01, -1.81640625e-01,
                1.38671875e-01, -3.10546875e-01, -9.71679688e-02,  1.31835938e-01,
               -1.16210938e-01,  7.03125000e-02,  2.85156250e-01,  3.51562500e-02,
               -1.01562500e-01, -3.75976562e-02,  1.41601562e-01,  1.42578125e-01,
               -5.68847656e-02,  2.65625000e-01, -2.09960938e-01,  9.64355469e-03,
               -6.68945312e-02, -4.83398438e-02, -6.10351562e-02,  2.45117188e-01,
```

How to get from word to document?

```
import spacy

# Load the spacy model that we already installed in Chapter 2. This takes a few seconds.
%time nlp = spacy.load('en_core_web_md')
# process a sentence using the model
mydoc = nlp("Canada is a large country")
#Get a vector for individual words
#print(doc[0].vector) #vector for 'Canada', the first word in the text
print(mydoc.vector) #Averaged vector for the entire sentence
```

Out of vocabulary words

What do we do when we encounter a new word not seen in the training vocabulary?

- Leave it out of your feature vector calculation for the text (a common strategy)
- Randomly initialize a vector
- "Learn" a representation for a OOV word by randomly replacing words during training with a new word token.
- ....

A better way: Character level embeddings

- Character level embedding models deal with unknown words and sparsity in languages with rich morphology, by using subword models.
- E.g., fasttext open-source library, including pretrained embeddings for 157 languages, is available at https://fasttext.cc.

using FastText

Using a pre-trained model

```
import fasttext
model = fasttext.train_unsupervised('data/fil9', thread=4)
[model.get_word_vector(x) for x in
    ["asparagus", "pidgey", "yellow"]]
```

source

Visualizing Embeddings

- Visualizing embeddings is an important goal in helping understand, apply, and improve these models of word meaning.
- But how can we visualize a (for example) 300-dimensional vector?

Visualizing Embeddings

- Visualizing embeddings is an important goal in helping understand, apply, and improve these models of word meaning.
- But how can we visualize a (for example) 300-dimensional vector?
  1. Check the most similar words to a given word, using some measure of distance (cosine)
  2. (More common): t-sne, which can project high dimensions into lower dimensions, like we saw earlier with sentiment words example.

Code examples: Notebooks 9 and 10 in Practical NLP's Chapter 3 @Github

Bias and Embeddings

- In addition to their ability to learn word meaning from text, embeddings, also reproduce the implicit biases and stereotypes that were latent in the text.
- Some past research showed that the closest occupation to 'man' - 'computer programmer' + 'woman' in word2vec embeddings trained on news text is 'homemaker', and that the embeddings similarly suggest the analogy 'father' is to 'doctor' as 'mother' is to 'nurse'.
- Recent research focuses on ways to try to remove this kind of biases in embedding representations.

A summary so far:

- These embedding methods are very fast, efficient to train, and easily accessible in terms of both code and pre-trained models.
- Drawback: Word2vec like embeddings are static embeddings i.e., there is a fixed embedding for each word in the vocabulary.
  - What about the multiple senses of a word?
  - Will a word's representation remain same irrespective of the context of its use?

Solution: Contextual Word Embeddings

- Idea: A word's embedding representation need not necessarily capture all possible uses of the word. It should capture only what it means in that context.
- How?: One approach - train a neural network, that takes a word, looks at its left/right context in the sentence, and estimate a vector representation of this context.

Solution: Contextual Word Embeddings

- Idea: A word's embedding representation need not necessarily capture all possible uses of the word. It should capture only what it means in that context.
- How?: One approach - train a neural network, that takes a word, looks at its left/right context in the sentence, and estimate a vector representation of this context.
- So, we can start with word2vec like embeddings, and then let the neural network "tune" these representations based on context too!

Solution: Contextual Word Embeddings

- Idea: A word's embedding representation need not necessarily capture all possible uses of the word. It should capture only what it means in that context.
- How?: One approach - train a neural network, that takes a word, looks at its left/right context in the sentence, and estimate a vector representation of this context.
- So, we can start with word2vec like embeddings, and then let the neural network "tune" these representations based on context too!
- Depending on how this context is learnt, many different contextual word embeddings were proposed in the past 5 years.
- BERT, the now famous NLP model, is based on this idea of contextual embeddings.

How do we get these representations?

Language Model pre-training

- Language modeling is a task of predicting the probability of a given sequence of words in a language.
- It is useful in a range of application scenarios, from speech recognition to spelling correction.

How do we get these representations?

Language Model pre-training

- Language modeling is a task of predicting the probability of a given sequence of words in a language.
- It is useful in a range of application scenarios, from speech recognition to spelling correction.
- In recent years, neural language models became popular in NLP.
- Why?:

How do we get these representations?

Language Model pre-training

- Language modeling is a task of predicting the probability of a given sequence of words in a language.
- It is useful in a range of application scenarios, from speech recognition to spelling correction.
- In recent years, neural language models became popular in NLP.
- Why?: they just need large amounts of text (which is now available for several languages), and learn to predict this probability in an unsupervised manner.
- In this process, they also learn effective contextual representations of text.

BERT-1

Bidirectional Encoder Representations from Transformers

- BERT is perhaps the most popular language model in NLP right now.
- learns contextual representations of text by training on "unsupervised" prediction tasks - masked language modeling, and next sentence prediction.

BERT-1

Bidirectional Encoder Representations from Transformers

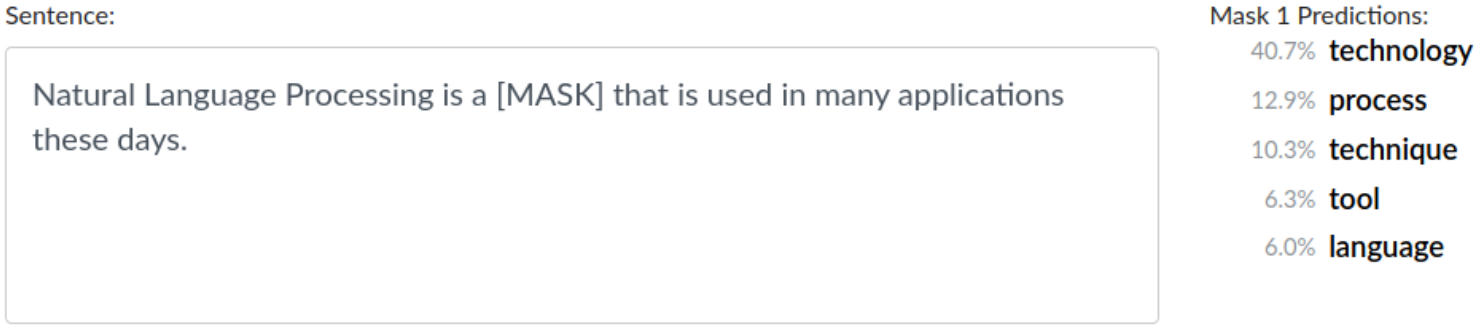
- BERT is perhaps the most popular language model in NLP right now.
- learns contextual representations of text by training on "unsupervised" prediction tasks - masked language modeling, and next sentence prediction.
- "Masked language modeling". It is similar to a fill-in-the-blank task, where a model learns to predict what the [MASK] token is, based on the context words surrounding it.
- How does learning like this capture anything beyond one sentence? next sentence prediction task: learning to predict the next sentence, given a sentence.

BERT-1

Bidirectional Encoder Representations from Transformers

- BERT is perhaps the most popular language model in NLP right now.
- learns contextual representations of text by training on "unsupervised" prediction tasks - masked language modeling, and next sentence prediction.
- "Masked language modeling". It is similar to a fill-in-the-blank task, where a model learns to predict what the [MASK] token is, based on the context words surrounding it.
- How does learning like this capture anything beyond one sentence? next sentence prediction task: learning to predict the next sentence, given a sentence.
- These are shown to be better representations of text than previous methods in various use cases.

BERT -Masked Language Model Demo



https://demo.allennlp.org/masked-lm

What do they learn?

- Depends on the model's neural network architecture.
- Various research studies showed that such contextual representations learn different kinds of syntactic properties (even though they are not explicitly trained for that).
- "probing" models to understand what they learn is an active area of research in NLP now.

How can we use these learned representations?

- As feature extractors: instead of using static embeddings, or BOW/TF-IDF or hand crafted representations, these representations can directly be used as features for downstream tasks such as text classification, information extraction etc.

How can we use these learned representations?

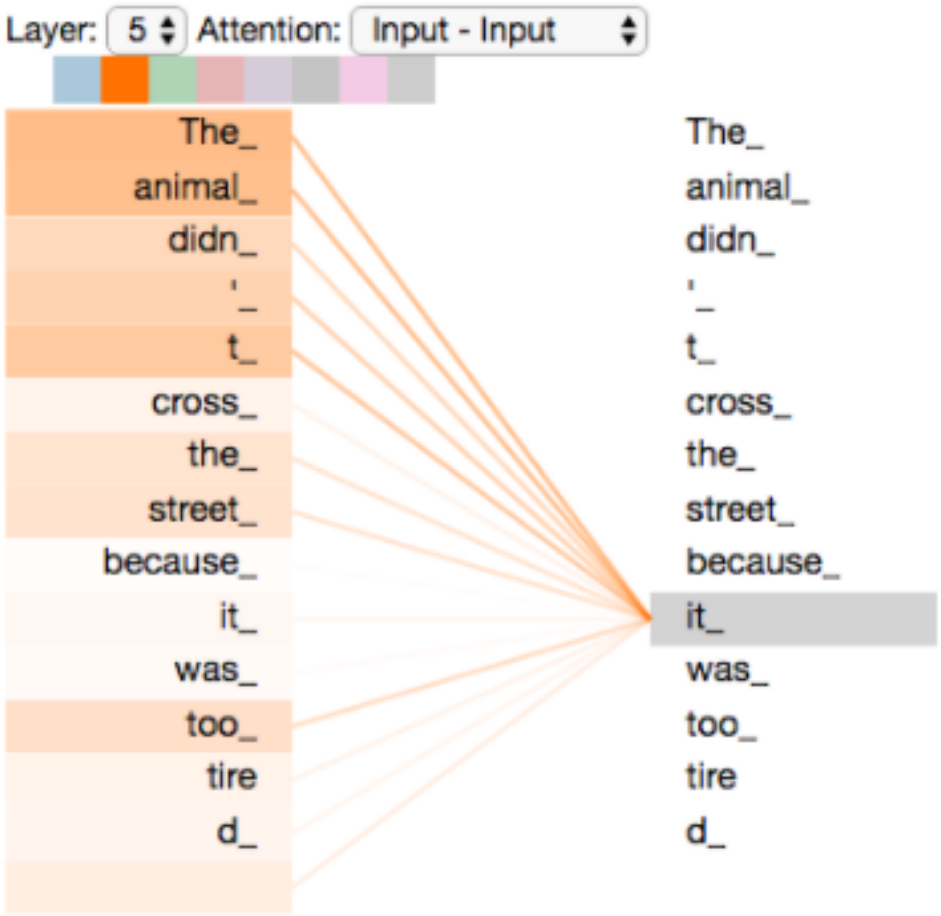
- As feature extractors: instead of using static embeddings, or BOW/TF-IDF or hand crafted representations, these representations can directly be used as features for downstream tasks such as text classification, information extraction etc.
- For fine-tuning: start with these representations (which were learnt from training on language modeling tasks) and let a neural network "tune" them to suit the current task (e.g., text classification).

What is fine-tuning?

- The representations learned from training a BERT model can be used as inputs/features to any other task (e.g., text classification)
- When we are using a neural network to learn this task, it transforms these input representations to suit that particular task.
- The pre-trained model's weights are then altered ("fine-tuned") while training for the task i.e., this pre-trained architecture is then "retrained" to suit this task.

What is happening during fine-tuning?

Let us start with "before" fine-tuning



Inferring association between tokens using attention. source: http://jalammar.github.io/illustrated-transformer/

What happens during fine-tuning?

A case study of using BERT to fine tune "aspect based sentiment analysis"

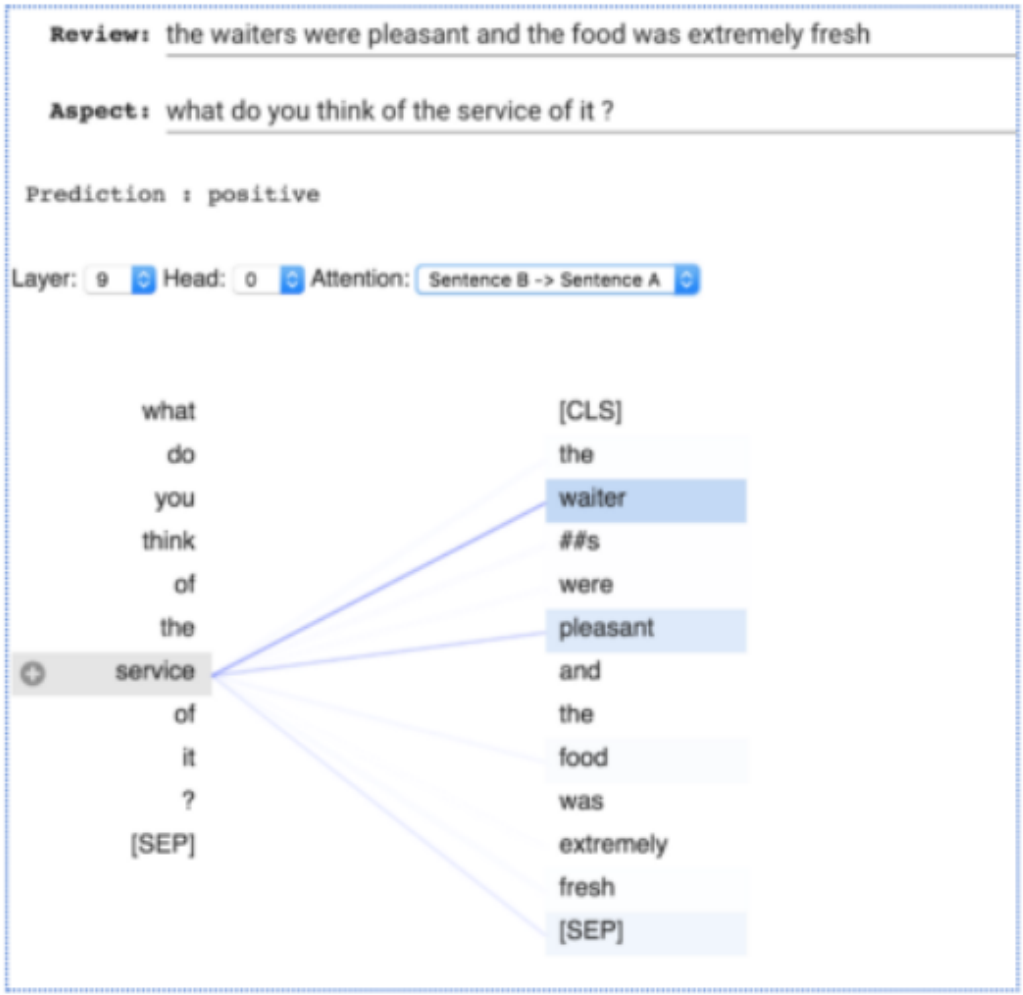
Review Sentence	Price	Food
The restaurant was too expensive	Negative	None
The restaurant was expensive, but the menu was great	Negative	Positive

Review Sentence	Question	Sentiment
The restaurant was too expensive	What do you think of the price of it ?.	Negative
The restaurant was too expensive	What do you think of the food of it ?.	None
The restaurant was expensive, but the menu was great	What do you think of the price of it ?.	Negative
The restaurant was expensive, but the menu was great	What do you think of the food of it ?.	Positive

Aspect-based sentiment analysis as QA — https://arxiv.org/pdf/1903.09588v1.pdf

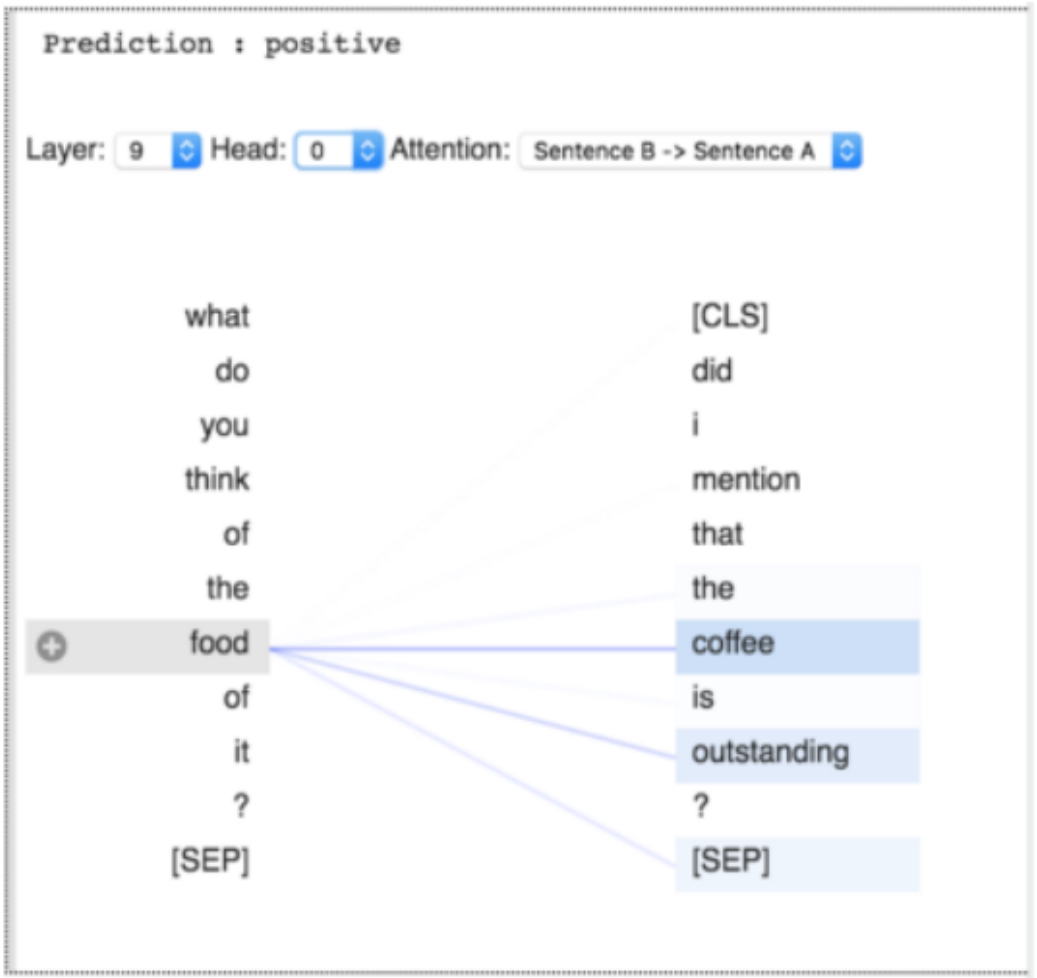
"What does a fine-tuned bert model look at?"

What happens during fine-tuning?



"What does a fine-tuned bert model look at?"

What happens during fine-tuning?



"What does a fine-tuned bert model look at?"

Does this really work in practice?

A small exercise

Visit https://huggingface.co/tasks, pick a task, and browse some model demos there. Check out how they work for a few minutes. (10 minutes)

Share your observations.

Beyond the original BERT

- Improvements to model architecture (e.g., RoBERTa)
- Language specific BERT models (e.g., CamemBERT)
- Multilingual BERT models (e.g., mBERT)
- Compressed BERT (e.g., DistillBERT)

.....

Things to keep in mind

- It is expensive and resource intensive to train your own BERT or any other such language model.
- "Training GPT-3 would cost over $4.6M using a Tesla V100 cloud instance." (source)
- Using the pre-trained model + fine tuning is how it is used in general purpose cases.

Resources for learning further:

- "Embeddings in NLP" book and recent (Dec 2020) tutorial at COLING 2020 (video is in the website)
- Chapters 6, 7 in "Speech and Language Processing" 3rdEdition
- "Illustrated *" posts by Jay Alammar
- A Primer in BERTology: What we know about how BERT works

Applying NLP methods for Economics research - some notes

What is different about economics?

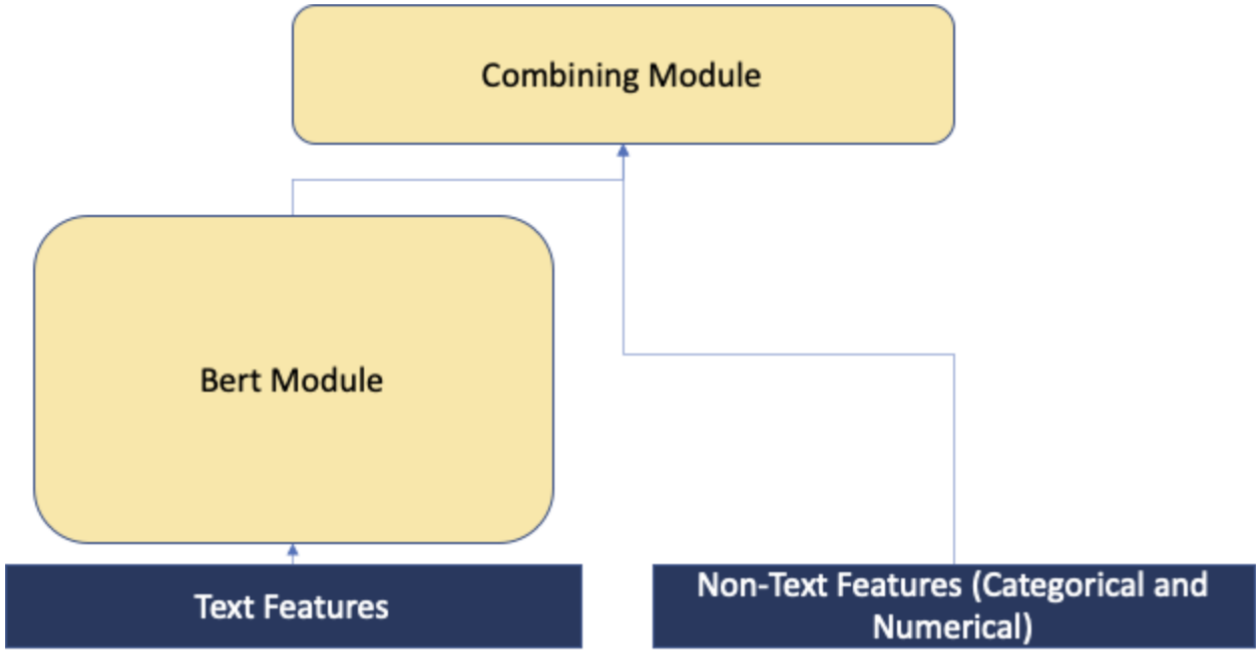
- Text is secondary information
- Often, there is a rich set of structured information available, which needs to be combined with this.
- e.g., information about the product, reviewer profile etc.
- How do you incorporate text features with existing data about the problem (e.g., other characteristics of the data, other information)?

Note: Other application areas such as clinical NLP also have this issue.

How can we approach this?

1. Concatenate them all into one vector (does not consider that the two sets of features don't carry same weight)
2. Use a model ensemble - one model with text features alone, one with other features alone, and weigh their predictions somehow.
3. More recent solution: multimodal transformers

Multimodal transformers



High level diagram of multimodal-transformers. The adaptation of transformers to incorporate data is all contained in the combining module.

Combining Module

A blog post by this team, with a case study on a clothing products recommendation dataset with text and numeric/categorical features.

Causal Inference and NLP

- NLP methods traditionally are correlation based, and the application scenarios also typically did not demand causal inference.

Causal Inference and NLP

- NLP methods traditionally are correlation based, and the application scenarios also typically did not demand causal inference.
- Yet, when used in some other disciplines (such as economics), it becomes important to figure out how to answer causal questions with textual data.

Causal Inference and NLP

- NLP methods traditionally are correlation based, and the application scenarios also typically did not demand causal inference.
- Yet, when used in some other disciplines (such as economics), it becomes important to figure out how to answer causal questions with textual data.
- This is a recent area of interest in NLP (e.g., 2021 Causal inference and NLP workshop)
- Archived proceedings from this workshop are open access here.

Rest of today's class

- Python notebook, looking into some existing NLP tools.

Next class

- Talking more in detail about two NLP tasks - text classification and topic modeling
- Reason: These two seem to be common NLP tasks in the economics papers I came across.
- Format: Lecture + Notebook

Now, let us move to notebooks again!