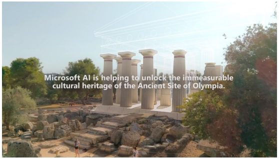
L98: Introduction to Computational Semantics

Lecture 2: Word Sense Disambiguation Algorithms

Simone Teufel and Weiwei Sun

Natural Language and Information Processing Research Group
Department of Computer Science and Technology
University of Cambridge

Lent 2021/22



Lecture 2: Word Sense Disambiguation Algorithms

1. WSD, the task
2. Lesk
3. Yarowsky
4. Supervised WSD
5. Neuralising the old

WSD, the Task

Word Sense Disambiguation

Helps in various NLP tasks:

- Machine Translation
- Question Answering
- Information Retrieval
- Text Classification

What counts as "one sense"?

- Task-specific senses
- dictionary-defined senses.

Sense-tagged corpora exist, e.g., SemCor

- 186 texts with all open class words WN synset tagged (192,639)
- 166 texts with all verbs WN synset tagged (41,497)

Types of Algorithms for WSD

- Supervised
  - range of classification algorithms, cf. end of lecture
- Unsupervised
  - Dictionary glosses (Lesk)
  - Lexical chains (Barzilay and Elhadad)
  - Graph properties of WN graph (Navigli and Lapata)
- Semi-supervised
  - Bootstrapping of context words (Yarowsky)
  - Active Learning
- Word Sense Induction
  - Always fully unsupervised
  - Typically, clustering-based

The Lesk Algorithm

Idea behind the Original Lesk: Mutual disambiguation

Typically there is more than one ambiguous word in the sentence.

Example

- *Several rare ferns grow on the steep banks of the burn where it runs into the lake.*

Ambiguous: *rare, steep, bank, burn, run*

But: humans do not perceive this sentence as ambiguous at all.

Algorithm 1

```
function SIMPLIFIED_LESK(word, sentence) returns best sense of word
  best-sense := most frequent sense for word
  max-overlap := 0
  context := set of words in sentence
  foreach sense in senses of word do
    signature := set of words in gloss and examples of sense
    overlap := COMPUTE_OVERLAP(signature, context)
    if overlap > max-overlap then
      max-overlap := overlap
      best-sense := sense
    end
  return best-sense
```

- Algorithm chooses the sense of target word whose gloss shares most words with sentence
- `COMPUTE_OVERLAP` returns the number of words in common between two sets, ignoring function words or other words on a stop list.

Solution to Pre-lecture exercise

$\mathbf{1} \rightarrow \mathbf{D}$
$\mathbf{2} \rightarrow \mathbf{A}$
$\mathbf{3} \rightarrow \mathbf{G}$
$\mathbf{4} \rightarrow \mathbf{I}$
$\mathbf{5} \rightarrow \mathbf{C}$
$\mathbf{6} \rightarrow \mathbf{B}$
$\mathbf{7} \rightarrow \mathbf{F}$
$\mathbf{8} \rightarrow \mathbf{H}$
$\mathbf{9} \rightarrow \mathbf{E}$

And this is why

[home/1, place] : *(where you live at a particular time) "deliver the package to my home"; "he doesn't have a home to go to"; "your place or mine?"*

[home/2, dwelling, domicile, abode, habitation, dwelling house]: *(housing that someone is living in) "he built a modest dwelling near the pond"; "they raise money to provide homes for the homeless"*

[home/3]: *(the country or state or city where you live) "Canadian tariffs enabled United States lumber companies to raise prices at home"; "his home is New Jersey"*

[home/4, home plate, home base, plate] : *((baseball) base consisting of a rubber slab where the batter stands; it must be touched by a base runner in order to score) "he ruled that the runner failed to touch home"*

[home/5, base]: *(the place where you are stationed and from which missions start and end)*

[home/6]: *(place where something began and flourished) "the United States is the home of basketball"*

[home/7]: *(an environment offering affection and security) "home is where the heart is"; "he grew up in a good Christian home"; "there's no place like home"*

[home/8, family, household, house, menage]: *(a social unit living together) "he moved his family to Virginia"; "It was a good Christian household"; "I waited until the whole house was asleep"; "the teacher asked how many people made up his home"; "the family refused to accept his will"*

[home/9, nursing home, rest home]: *(an institution where people are cared for) "a home for the elderly"*

Example: Disambiguation of *bank*

Context: *The bank can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.*

bank/1	(a financial institution that accepts deposits and channels the money into lending activities) *"he cashed a check at the bank"*, *"that bank holds the mortgage on my home"*
bank/2	(sloping land (especially the slope beside a body of water)) *"they pulled the canoe up on the bank"*, *"he sat on the bank of the river and watched the currents"*

- Sense *bank/1* has two (non-stop) words overlapping with the context (*deposits* and *mortgage*)
- Sense *bank/2* has zero, so sense *bank/1* is chosen.

Algorithm 2

```
function LESK(word, sentence) returns best sense of word
  best-sense := most frequent sense for word
  max-overlap := 0
  context := set of words in sentence
  foreach sense in senses of word do
    signature := set of words in gloss of sense
    foreach context-word in context do
      foreach context-sense of context_word do
        context_signature := set of words in gloss of context_sense
        overlap := COMPUTE_OVERLAP(signature, context_signature)
        if overlap > max-overlap then
          max-overlap := overlap
          best-sense := sense
      end
    end
  end
  return best-sense
```

- Compare each target word's signature (sense-related words) with each of the context words' signatures.
- sense–sense comparison

Example: Disambiguation of *cone* and *pine*

Context: *pine cone*

pine/1	kinds of evergreen tree with needle-shaped leaves
pine/2	waste away through sorrow or illness

cone/1	solid body which narrows to a point
cone/2	something of this shape whether solid or hollow
cone/3	fruit of a certain evergreen tree

cone/3 and *pine/1* are selected:

- overlap for entries *pine/1* and *cone/3* (*evergreen* and *tree*)
- no overlap in other entries

Two "Lesk" algorithms

- Algorithm II is the one that was first published
- It is now called the Original Lesk (1986) Algorithm.
- In almost all situations it is beaten by Algorithm I ("Simplified Lesk"), due to Kilgarriff and Rosenzweig (2000)
- "Corpus" version of Algorithm I additionally expands glosses by all known contexts of that sense from SEMCOR.

Intrinsic evaluation

Sense accuracy: percentage of words where the system tag is identical to gold standard tag

How can we get annotated material cheaply?

- Pseudo-words
  - create artificial corpus by conflating unrelated words
  - example: replace all occurrences of *banana* and *door* with *banana-door*
- Multi-lingual parallel corpora
  - translated texts aligned at the sentence level
  - translation indicates sense

Competitive evaluations exist

- SENSEVAL; annotated corpora in many languages
- "Lexical Sample" Task for supervised WSD
- "All-word" Task for unsupervised WSD (SemCor corpus)

Baselines for supervised WSD

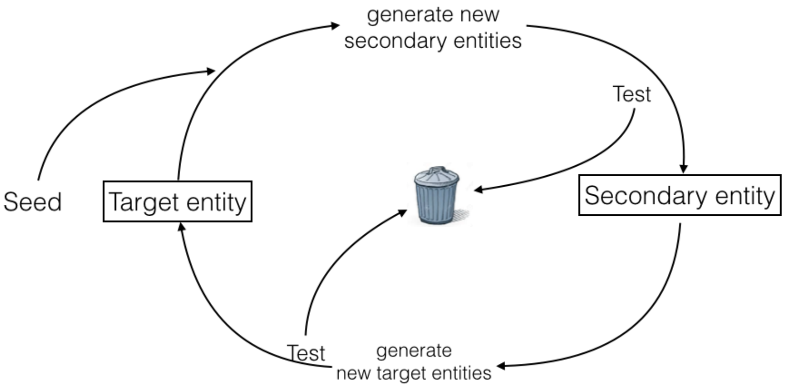
- First (most frequent) sense
- LeskCorpus (Simplified, weighted Lesk, with all the words in the labeled SEMEVAL corpus sentences for a word sense added to the signature for that sense)
- LeskCorpus is the best-performing of all symbolic Lesk variants, was used for a long time (Kilgarriff and Rosenzweig, 2000; Vasilescu et al., 2004)
- Nowadays, embedded version of Lesk is used; called 1-NN

The Yarowsky Algorithm

Semi-supervision and bootstrapping

- Baron Münchhausen, the famous lier.
- "I pulled myself out of the swamp, by pulling on my own hair"
- Term "bootstrapping" co-opted by Machine Learning
- Weakly supervised (semi-supervised): use only few labelled training examples
- Also called "seed" examples

Bootstrapping principle



Semi-supervised WSD by bootstrapping

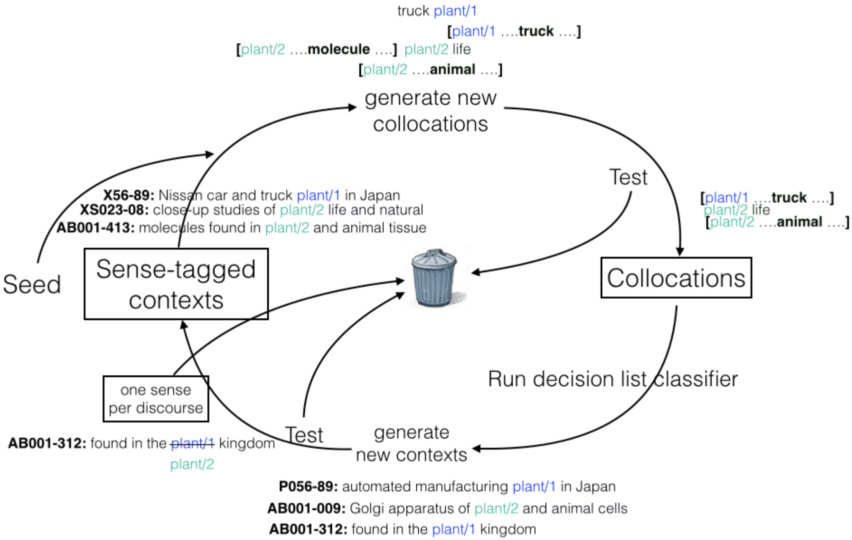
- The Yarowsky algorithm is an example of a bootstrapping algorithm
- That means it only requires a small amount of annotated data.
- However, many such algorithms use a large amount of non-annotated corpus material.
- This is an advantage, because hand-annotation is expensive.
- The algorithm steps:
    - It starts with a small seed set, trains a classifier on it, and then applies it to the whole data set (bootstrapping);
    - Reliable examples are kept, and the classifier is re-trained.

Yarowsky's algorithm

Yarowsky's (1995) algorithm uses two powerful heuristics for WSD:

- **One sense per collocation:** nearby words provide clues to the sense of the target word, conditional on distance, order, syntactic relationship.
  - The algorithm uses this to find good features
- **One sense per discourse:** the sense of a target words is consistent within a given document.
  - In 50% of all documents, the target word occurs more than once.
  - In 98% of these cases, they have the same sense.

Bootstrapping: Yarowsky



Seed set

Step 1: Extract all instances of a polysemous or homonymous word.

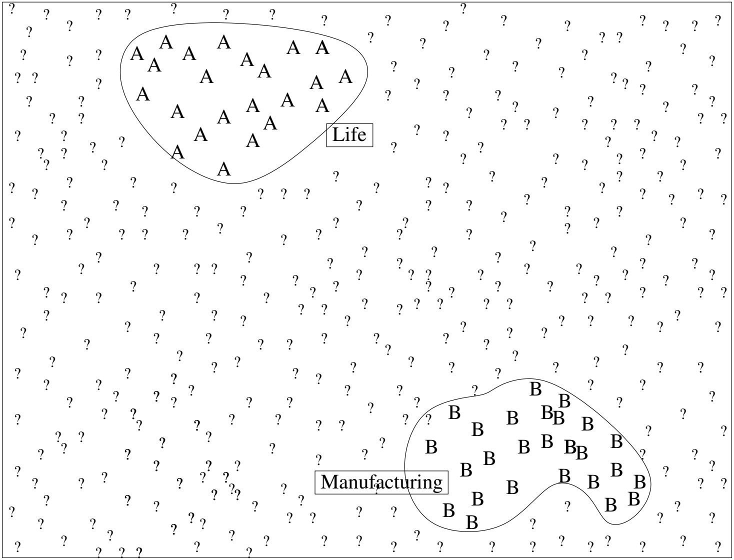
Step 2: Generate a seed set of labeled examples:

- either by manually labeling them;
- or by using a reliable heuristic.

Example: target word *plant*: As seed set take all instances of

- *plant life* (sense A) and
- *manufacturing plant* (sense B).

Figures and tables from now on: taken from Yarowsky (1995).


Life
Manufacturing

Classification

Step 3a: Train classifier on the seed set.

Step 3b: Apply classifier to the entire sample set. Add those examples that are classified reliably (probability above a threshold) to the seed set.

Yarowsky uses a **decision list** classifier:

- rules of the form: collocation $\rightarrow$ sense
- rules are ordered by log-likelihood:

$$\log \frac{P(sense_A | collocation_i)}{P(sense_B | collocation_i)}$$

- Classification is based on the first rule that applies.

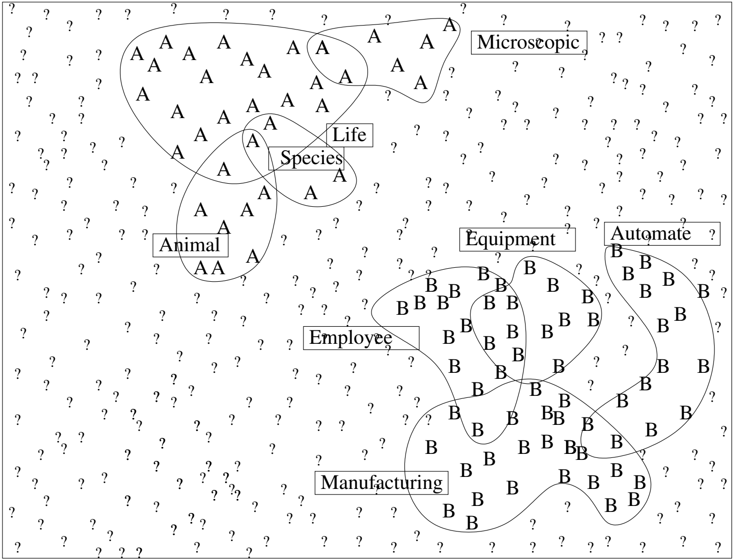
Classification: **initial** decision list for *plant*

LogL	Collocation	Sense
8.10	*plant* life	$\rightarrow$ A
7.58	manufacturing *plant*	$\rightarrow$ B
7.39	life (within $\pm$2-10 words)	$\rightarrow$ A
7.20	manufacturing (in $\pm$ 2-10 words)	$\rightarrow$ B
6.27	animal (within $\pm$2-10 words)	$\rightarrow$ A
4.70	equipment (within $\pm$2-10 words)	$\rightarrow$ B
4.39	employee (within $\pm$2-10 words)	$\rightarrow$ B
4.30	assembly *plant*	$\rightarrow$ B
4.10	*plant* closure	$\rightarrow$ B
3.52	*plant* species	$\rightarrow$ A
3.48	automate (within $\pm$2-10 words)	$\rightarrow$ B
3.45	microscopic *plant*	$\rightarrow$ A
	...	

Classification: **final** decision list for *plant*

LogL	Collocation	Sense
10.12	*plant* growth	$\rightarrow$ A
9.68	car (within $\pm$2-10 words)	$\rightarrow$ B
9.64	*plant* height	$\rightarrow$ A
9.61	union (in $\pm$ 2-10 words)	$\rightarrow$ B
9.54	equipment (within $\pm$ 2-10 words)	$\rightarrow$ B
9.51	assembly *plant*	$\rightarrow$ B
9.50	nuclear *plant*	$\rightarrow$ B
9.31	flower (within $\pm$2-10 words)	$\rightarrow$ A
9.24	job (within $\pm$2-10 words)	$\rightarrow$ B
9.03	fruit (within $\pm$2-10 words)	$\rightarrow$ A
9.02	*plant* species	$\rightarrow$ A
	...	

Classification



One sense per discourse

Step 3c: Use one-sense-per-discourse constraint to **expand** newly classified examples:

- If several examples in one document have already been annotated as sense A, then extend this to all examples of the word in the rest of the document.
- This can bring in new collocations, and even correct erroneously labeled examples.

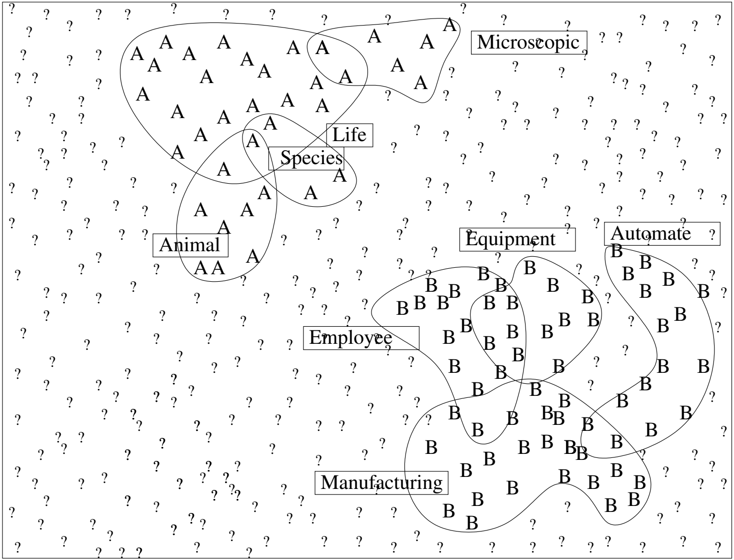
One sense per discourse

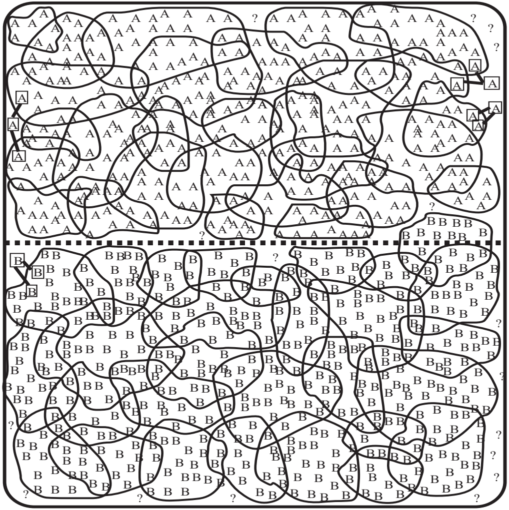
Step 3c: Use one-sense-per-discourse constraint to **filter** newly classified examples:

- If you detect a contradiction in one document, one of two cases applies:
  - Either your classifier was wrong on at least one of the examples $\rightarrow$ remove the collocation that resulted in this error
  - It's one of the rare cases where really two different senses do occur in a document $\rightarrow$ Do not use this document for training
- We don't really know which case applies, but we can use the confidence of the classifier as an approximation.

Step 3d: repeat Steps 3a–d.

Classification





Generalization

Step 4: Algorithm converges on a stable residual set (remaining unlabeled instances):

- most training examples will now exhibit multiple collocations indicative of the same sense;
- decision list procedure uses only the most reliable rule, not a combination of rules.

Step 5: The final classifier can now be applied to unseen data.

Discussion of Yarowsky

Strengths:

- simple algorithm that uses only minimal features (words in the context of the target word);
- minimal effort required to create seed set;
- does not rely on dictionary or other external knowledge.

Weaknesses:

- uses very simple classifier (but could replace it with a more state-of-the-art one);
- not fully unsupervised: requires seed data;
- does not make use of the structure of a possibly existing dictionary (the sense inventory).

Supervised WSD

Supervised WSD

- In Supervised WSD, words in the training data are labelled with their senses:
  - She pays 3% interest/INTEREST-MONEY on the loan.
  - He showed a lot of interest/INTEREST-CURIOSITY in the painting.
- You define features that (you hope) will indicate one sense over another
- Train a statistical model that predicts the correct sense given the features
- Classifier is trained for each target word separately

Features for supervised WSD

An electric guitar and **bass** *player stand off to one side, not really part of the scene, just as a sort of nod to gringo expectations perhaps.*

- Collocational feature: (directly neighbouring words in specific positions)
  [w_{i-2}, POS_{i-2}, w_{i-1}, POS_{i-1}, w_{i+1}, POS_{i+1}, w_{i+2}, POS_{i+2}]
  [guitar, NN, and, CC, player, NN, stand, VB]
- Bag of Words feature: (any content words in a 50 word window)
  12 most frequent content words from *bass* collection: [*fishing, big, sound, player, fly, rod, pound, double, runs, playing, guitar, band*]
  $\rightarrow$ [0,0,0,1,0,0,0,0,0,0,1,0]

Naive Bayes

- Goal: choose the best sense $\widehat{s}$ out of the set of possible senses S for an input vector $\overrightarrow{F}$:

$$\widehat{s} = \arg\max_{s \in S} P(s|\overrightarrow{F})$$

- It is difficult to collect statistics for this equation directly.
- Rewrite it using Bayes' rule:

$$\widehat{s} = \arg\max_{s \in S} = \frac{P(\overrightarrow{F}|s)P(s)}{P(\overrightarrow{F})}$$

- Drop $P(\overrightarrow{F})$ – it is a constant factor in $\arg\max$
- Assume that F_i are independent:

$$P(\overrightarrow{F}|s) \approx \prod_{n}^{j=1} P(F_i|s)$$

Naive Bayesian classifier

- Naive Bayes Classifier:

$$\widehat{s} = \arg\max_{s \in S} P(s) \prod_{n}^{j=1} P(F_i|s)$$

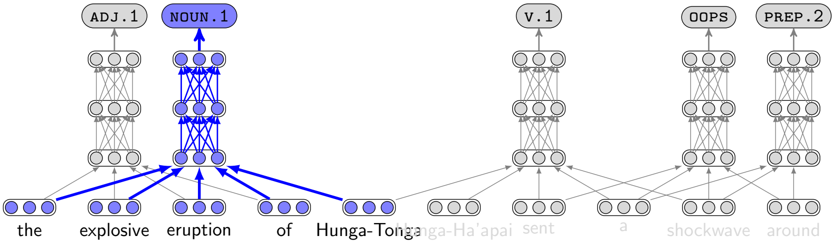
- Parameter Estimation (Max. likelihood):
  - How likely is sense s_i for word form w_j?

$$P(s_i) = \frac{count(s_i, w_j)}{count(w_j)}$$

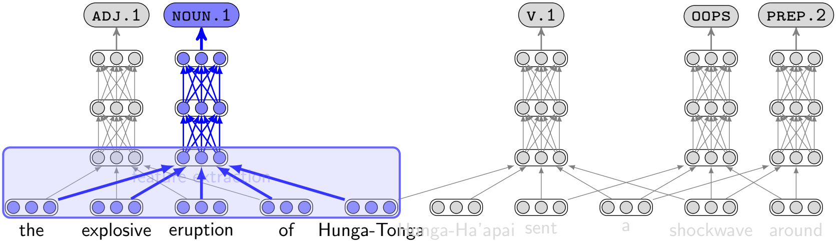
  - How likely is feature f_j given sense s_i?

$$P(F_j|s_i) = \frac{count(s_i, F_j)}{count(s_i)}$$

Classification based on dense word representations

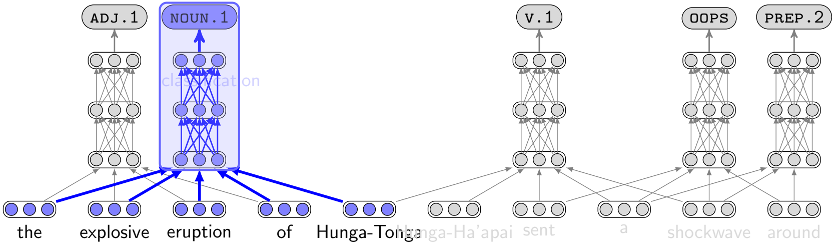


Classification based on dense word representations



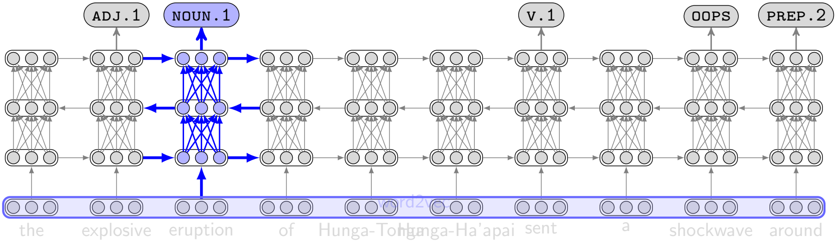
- manually defined features
- static word embeddings: `word2vec`, fastText, GloVe etc

Classification based on dense word representations



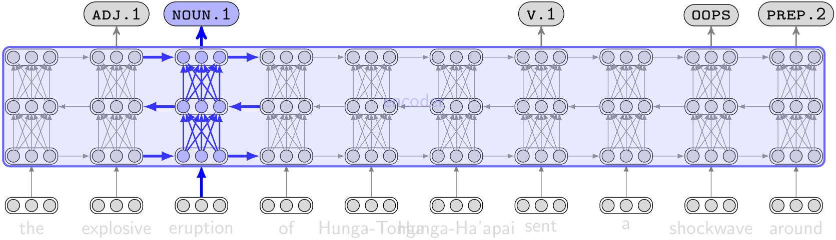
- manually defined features
- static word embeddings: `word2vec`, fastText, GloVe etc
- classifier: Multi-layer perceptron, etc

+a sentence encoder



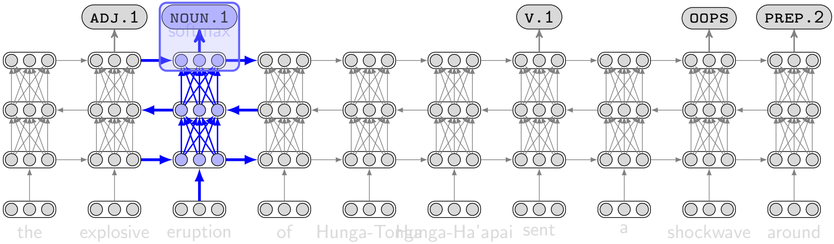
- static word embeddings: e.g., `word2vec`, etc ▷ based on word types

+a sentence encoder



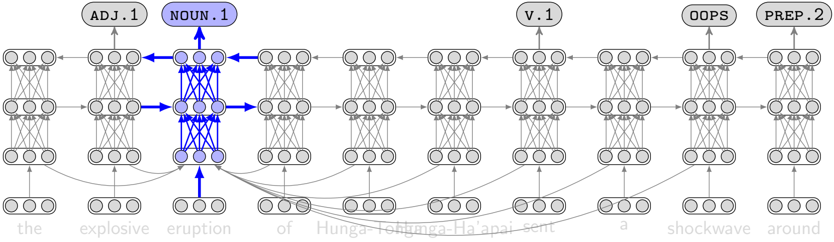
- static word embeddings: e.g., `word2vec`, etc ▷based on word types
- encoder: e.g. LSTM, transformer, etc

+a sentence encoder



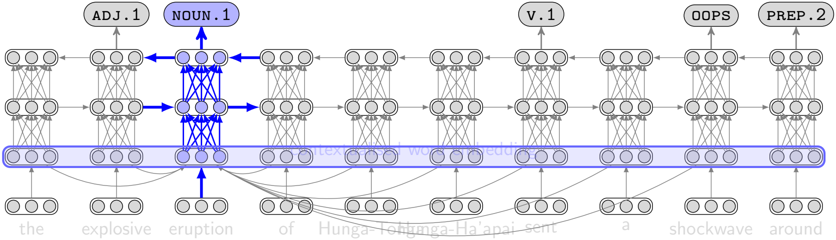
- static word embeddings: e.g., `word2vec`, etc ▷based on word types
- encoder: e.g. LSTM, transformer, etc
- classifier: `softmax` layer, etc

+contextualised word embeddings



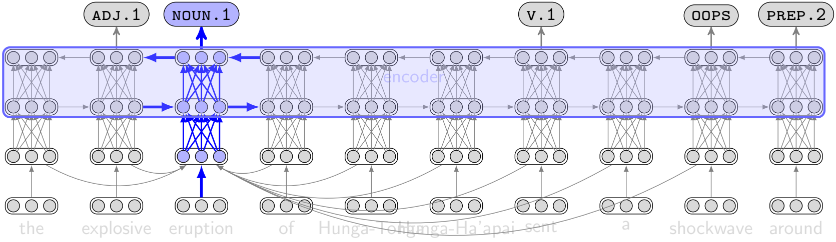
- word vectors: e.g., `word2vec`, etc

+contextualised word embeddings



- word vectors: e.g., `word2vec`, etc
- contextualised word embeddings, e.g. ELMo, BERT, etc

+contextualised word embeddings



- word vectors: e.g., `word2vec`, etc
- contextualised word embeddings, e.g. ELMo, BERT, etc
- encoder: e.g. LSTM, transformer, etc

Neuralising the Old

1-NN: Lesk with Sense embeddings

- Empirical results: Supervised WSD is better when a good amount of annotation is available.

⇒ The relative lack of annotation is a major limitation.

```
function SIMPLIFIED_LESK(word, sentence)
  best-sense := most frequent sense for word
  max-overlap := 0
  cntxt := set of words in sentence                    →EMBED(sentence)
  foreach sense in senses of word do
    sgn := set of words in gloss and ...              →EMBED(gloss,...)
    overlap := COMPUTE_OVERLAP(sgn, cntxt)             →SIM(sgn, cntxt)
    if overlap > max-overlap then
      max-overlap := overlap
      best-sense := sense
    end
  return best-sense
```

bag of words → vectors

Summary of lecture

- The **Lesk** algorithm uses overlap between context and glosses.
  - Idea: mutual disambiguation
- The **Yarowsky** algorithm uses bootstrapping and two key heuristics:
  - one sense per collocation;
  - one sense per discourse;
- **Supervised WSD** learns from context and uses ML to learn the best representations for senses (e.g. neurally).
- Fully unsupervised WSD can also be seen as **Word sense induction**

Reading

- Jurasfky and Martin, chapter 20.1-20.4.
- **Lesk** (1986): Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In SIGDOC '86, ACM.
- **Yarowsky** (1995): Unsupervised Word Sense Disambiguation rivaling Supervised Methods. Proceedings of the ACL.
- **Raganato, Camacho-Collados and Navigli** (2017): Word Sense Disambiguation: A Unified Evaluation Framework and Empirical Comparison.
- **Loureiro and Jorge (2019)**: Language Modelling Makes Sense: Propagating Representations through WordNet for Full-Coverage Word Sense Disambiguation.