CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computer Science

http://www.cit.ie

Natural Language Processing

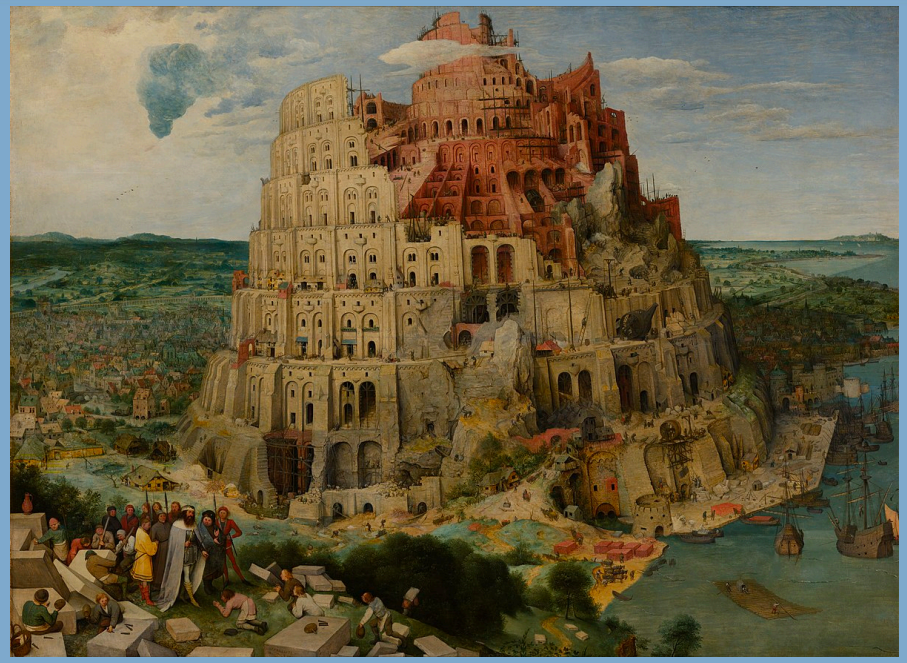
Week4: Language Models Smoothing and Evaluation

Dr. Haithem Afli

Haithem. afli@cit.ie

@AfliHaithem

2020/2021



Content

1. What is a Language Model?
2. N-gram Language Model
3. **Smoothing**
4. Evaluation
5. Management of Large Language Models
6. Other Approaches

Language Model Smoothing

- Smoothing in language modelling is the process by which unseen events are assigned a non-zero probability

- This is achieved by some combination of
  - Count adjustment
  - Interpolation
  - Back-off

Count Adjustment

count(tea) = 10

count(tea cup) = 3

count(tea drinker) = 3

count(tea time) = 4

$$p(x|tea)=\frac{count(tea\ x)}{count(tea)}$$

$$p(cup|tea) = .3$$
$$p(drinker|tea) = .4$$
$$p(time|tea) = .3$$

Count Adjustment

What happens when we want to estimate the probability of the phrase "$tea\ cosy$"?

Count Adjustment

- What happens when we want to estimate the probability of the phrase "$tea\ cosy$" ?

$$p(cosy|tea) = 0/10 = 0$$

Just because we haven't seen an event in our sample, it doesn't mean that it can't occur. The smaller our sample size, the less we trust counts derived from it.

Therefore, we need to assign $p(cosy|tea)$ a non-zero probability.

Count Adjustment

Remember that for our model to be sound

$$\sum_{x} p(x|tea) = 1$$

Count Adjustment

So, for $p(cosy|tea)$ not to be zero, we have to adjust downwards the probability of our seen events:

- $p(time|tea)$
- $p(cup|tea)$
- $p(drinker|tea)$

How do we do that?

Add-One Smoothing

For all possible events, add the count of one.

$$p = \frac{c+1}{n+v}$$

In case of n-gram language model:

$p = p(w_n|w_1, \ldots, w_{n-1})$

c = count of n-gram $(w_1, \ldots, w_{n-1}, w_n)$ in corpus

n = count of history $(w_1, \ldots, w_{n-1})$ in corpus

v = vocabulary size

Add-One Smoothing

Words in language= $\{tea, time, cup, drinker, cosy\}$

Vocabulary size =5

Observed events =

$tea\ time,\quad tea\ time,\quad tea\ time,\quad tea\ cup,$
$tea\ cup,\quad tea\ cup,\quad tea\ drinker,\quad tea\ drinker,$
$tea\ drinker,\quad tea\ drinker$

	tea	time	cup	drinker	cosy	SUM
tea	0	3	3	4	0	10
time	0	0	0	0	0	0
cup	0	0	0	0	0	0
drinker	0	0	0	0	0	0
cosy	0	0	0	0	0	0

Add-One Smoothing

	tea	time	cup	drinker	cosy	SUM
tea	0+1=1	3+1=4	3+1=4	4+1=5	0+1=1	15
time	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5
cup	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5
drinker	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5
cosy	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	5

$$p(cup|tea) = (3+1)/(10+5) = 4/15$$
$$p(time|tea) = (3+1)/(10+5) = 4/15$$
$$p(drinker|tea) = (4+1)/(10+5) = 5/15$$
$$p(cosy|tea) = (0+1)/(10+5) = 1/15$$
$$p(tea|tea) = (0+1)/(10+5) = 1/15$$

$$p(cup|cup) = 1/5$$
$$p(time|cup) = 1/5$$
$$p(drinker|cup) = 1/5$$
$$p(cosy|cup) = 1/5$$
$$p(tea|cup) = 1/5$$

Add-One Smoothing

$$p(tea\ cozy)$$

$$= p(tea)p(cozy|tea)$$

$$= \frac{10}{20} \times \frac{1}{15} = \frac{1}{30}$$

Here we use unigram language model to estimate $p(tea)$.

Deal with Sentence Boundaries

If we add sentence boundaries to observed events:

Observed events =

<s>*tea time*</s>, <s>*tea time*</s>, <s>*tea time*</s>, <s>*tea cup*</s>, <s>*tea cup*</s>, <s>*tea cup*</s>, <s>*tea drinker*<s>, <s>*tea drinker*</s>, <s>*tea drinker*</s>, <s>*tea drinker*</s>

	tea	time	cup	drinker	cosy	</s>	SUM
<s>	10	0	0	0	0	0	10
tea	0	3	3	4	0	0	10
time	0	0	0	0	0	3	0
cup	0	0	0	0	0	3	0
drinker	0	0	0	0	0	4	0
cosy	0	0	0	0	0	0	0

Deal with Sentence Boundaries

Add-one Smoothing:

	tea	time	cup	drinker	cosy	</s>	SUM
<s>	10+1=11	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	16
tea	0+1=1	3+1=4	3+1=4	4+1=5	0+1=1	0+1=1	16
time	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	3+1=4	9
cup	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	3+1=4	9
drinker	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	4+1=5	10
cosy	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	0+1=1	6

$$p(cup|tea) = (3+1)/(10+6) = 4/16$$
$$p(time|tea) = (3+1)/(10+6) = 4/16$$
$$p(drinker|tea) = (4+1)/(10+6) = 5/16$$
$$p(cosy|tea) = (0+1)/(10+6) = 1/16$$
$$p(tea|tea) = (0+1)/(10+6) = 1/16$$

$$p(cup|cup) = 1/6$$
$$p(time|cup) = 1/6$$
$$p(drinker|cup) = 1/6$$
$$p(cosy|cup) = 1/6$$
$$p(tea|cup) = 1/6$$

Deal with Sentence Boundaries

$$p(<s>tea\ cozy</s>)$$

$$= p\ (tea|<s>)p(cozy|tea)p(</s>|cozy)$$

$$= \frac{11}{16} \times \frac{1}{16} \times \frac{1}{6} = \frac{11}{1536}$$

Add-One Smoothing

But there are many more unseen n-grams than seen n-grams.

Example: Europarl 2-bigrams:

- 86,700 distinct words
- $86{,}700^2$ = 7,516,890,000 possible bigrams (v)
- but only about 30,000,000 words (and bigrams) in corpus

Seen n-grams are assigned too low a probability!

Add-α Smoothing

Instead of adding one to the count, we add a lower value α (<1)

$$p = \frac{c + \alpha}{n + \alpha v}$$

Smoothing: the story so far

- The smoothing techniques covered so far work by adjusting (lowering) the counts of seen n-grams so that unseen n-grams get assigned some of the probability mass
- Other techniques
  - Interpolation
  - Back-off
- These are based on the idea of combining language models of different orders

Discussion

Combining language models

- In given corpus, we may never observe
  - Turkish coffee drinkers
  - Turkish coffee eaters
- Both have count 0
- The smoothing methods covered so far will assign them the same probability

Combining language models

- In given corpus, we may never observe
  - Turkish coffee drinkers
  - Turkish coffee eaters
- Both have count 0
- The smoothing methods covered so far will assign them the same probability
- It would be useful to look at bigram frequencies
  - coffee drinkers
  - coffee eaters

Interpolation

Higher and lower order n-gram models have different strengths and weaknesses

- high-order n-grams
  - sensitive to more context
  - sparse counts
- low-order n-grams
  - very limited context
  - have robust counts

Interpolation

- Combine them

$$p(w_3|w_1, w_2) = \lambda_1\, p1(w_3) + \lambda_2\, p2(w_3|w_2) + \lambda_3\, p3(w\,3|w\,1, w\,2)$$

Where: $\lambda_1 + \lambda_2 + \lambda_3 = 1$

- Recursively

$$p_n^I(w_i|w_{i-n+1}, \ldots, w_{i-1}) = \lambda_{w_{i-n+1},\ldots,w_{i-1}}\; p_n(w_i|w_{i-n+1}, \ldots, w_{i-1}) + \\ +\; (1 - \lambda_{w_{i-n+1},\ldots,w_{i-1}})\; p_{n-1}^I(w_i|w_{i-n+2}, \ldots, w_{i-1})$$

Backoff

- Backoff is slightly different
- The basic idea is to trust the highest order language model that contains the n-gram

Backoff

- Given a trigram, bigram and unigram language model
  - if trigram count > 0:
    - Use it
  - else if bigram count > 0:
    - Use it
  - else:
    - Use the unigram count

Other Considerations
Diversity of Predicted Words

- Consider the English words *spite* and *constant*
  - both occur 993 times in Europarl corpus
  - only 9 different words follow *spite*
  - almost always followed by *of* (979 times), due to expression *in spite of*
  - 415 different words follow *constant*
  - most frequent: *and* (42 times), *concern* (27 times), *pressure* (26 times), but huge tail of singletons: 268 different words

Other Considerations
Diversity of Predicted Words

- More likely to see new bigram that starts with *constant* than *spite*
- Witten-Bell smoothing takes this into account

Witten-Bell smoothing

- Recursive interpolation

$$p_n^I(w_i|w_{i-n+1},...,w_{i-1}) = \lambda_{w_{i-n+1},...,w_{i-1}}\ p_n(w_i|w_{i-n+1},...,w_{i-1}) + \\ +\ (1-\lambda_{w_{i-n+1},...,w_{i-1}})\ p_{n-1}^I(w_i|w_{i-n+2},...,w_{i-1})$$

- Number of possible extensions of a history

$$N_{1+}(w_1,...,w_{n-1},\bullet) = |\{w_n : c(w_1,...,w_{n-1},w_n) > 0\}|$$

- Lambda parameters

$$1-\lambda_{w_1,...,w_{n-1}} = \frac{N_{1+}(w_1,...,w_{n-1},\bullet)}{N_{1+}(w_1,...,w_{n-1},\bullet) + \sum_{w_n} c(w_1,...,w_{n-1},w_n)}$$

Witten-Bell smoothing

- Example
  - Nb ocurrences spite and constant (993)
  - Words following spite (9) and constant (415)

$$1 - \lambda_{spite} = \frac{N_{1+}(\text{spite}, \bullet)}{N_{1+}(\text{spite}, \bullet) + \sum_{w_n} c(\text{spite}, w_n)}$$

$$= \square$$

$$1 - \lambda_{constant} = \frac{N_{1+}(\text{constant}, \bullet)}{N_{1+}(\text{constant}, \bullet) + \sum_{w_n} c(\text{constant}, w_n)}$$

$$= \square$$

Witten-Bell smoothing

- Example
  - No ocurrences spite and constant (993)
  - Words following spite (9) and constant (415)

$$1 - \lambda_{spite} = \frac{N_{1+}(\text{spite}, \bullet)}{N_{1+}(\text{spite}, \bullet) + \sum_{w_n} c(\text{spite}, w_n)}$$

$$= \frac{9}{9 + 993} = 0.00898$$

$$1 - \lambda_{constant} = \frac{N_{1+}(\text{constant}, \bullet)}{N_{1+}(\text{constant}, \bullet) + \sum_{w_n} c(\text{constant}, w_n)}$$

$$= \frac{415}{415 + 993} = 0.29474$$

Other Considerations

- Consider the word *York*
  - fairly frequent word in Europarl corpus, occurs 477 times
  - as frequent as *foods*, *indicates* and *providers*
  - in unigram language model: a respectable probability

Other Considerations

- However, it almost always directly follows *New* (473 times, out of 477)
- Recall: unigram model only used if the bigram model inconclusive
- *York* unlikely second word in unseen bigram
- In backoff unigram model, *York* should have low probability
- **Kneser-Ney** smoothing takes this into account

Content

1. What is a Language Model?
2. N-gram Language Model
3. Smoothing
4. **Evaluation**
5. Management of Large Language Models
6. Other Approaches

Evaluating Language Model

Best evaluation for comparing models A and B

- Put each model in a task
- ➢ Spelling corrector, speech recognizer, MT system
- Run the task, get an accuracy for A and for B How many misspelled words corrected properly
- ➢ How many words translated correctly
- Compare accuracy for A and B
- ➢ Time consuming; can take days or weeks!

Evaluating Language Model

- How well can we predict the next word?

Evaluating Language Models

The quality of a language model can be estimated using perplexity

$$PP = 2^{H(P_{LM})}$$

which in turn is based on the notion of cross-entropy

$$H(P_{LM}) = -\frac{1}{n}\log P_{LM}(w_1 \ldots w_n)$$

Evaluating Language Model

- The Shannon Game:
  - How well can we predict the next word?

    I always order pizza with cheese and ____

    The 33rd President of the US was ____

    I saw a ____

  - Unigrams are terrible at this game. (Why?)

mushrooms 0.1
pepperoni 0.1
anchovies 0.01
….
fried rice 0.0001
….
and 1e-100

- A better model of a text
  - is one which assigns a higher probability to the word that actually occurs

Example

- Say we wanted to measure how well a language model p_{LM} predicts the following sentence:

I would like to commend the rapporteur on his work.

A low perplexity indicates the probability distribution is good at predicting the sample.

Content

1. What is a Language Model?
2. N-gram Language Model
3. Smoothing
4. Evaluation
5. **Management of Large Language Models**
6. Other Approaches

Managing very large language models

- Millions to billions of words are easy to get (trillions of English words available on the web)
- But: huge language models do not fit into RAM

Delete singletons

Europarl data:

Order	Unique n-grams	Singletons
unigram	86,700	33,447 (38.6%)
bigram	1,948,935	1,132,844 (58.1%)
trigram	8,092,798	6,022,286 (74.4%)
4-gram	15,303,847	13,081,621 (85.5%)
5-gram	19,882,175	18,324,577 (92.2%)

Delete singletons of higher order n-grams

Other Techniques

- Reduce **vocabulary size** – e.g. mapping the numbers to the symbol NUM
- Store **on disk** and load only **what is necessary**

into memory

- **Filter**: keep only words that are in the translation options being explored by the decoder
- **Quantisation**: use fewer bits to store probabilities

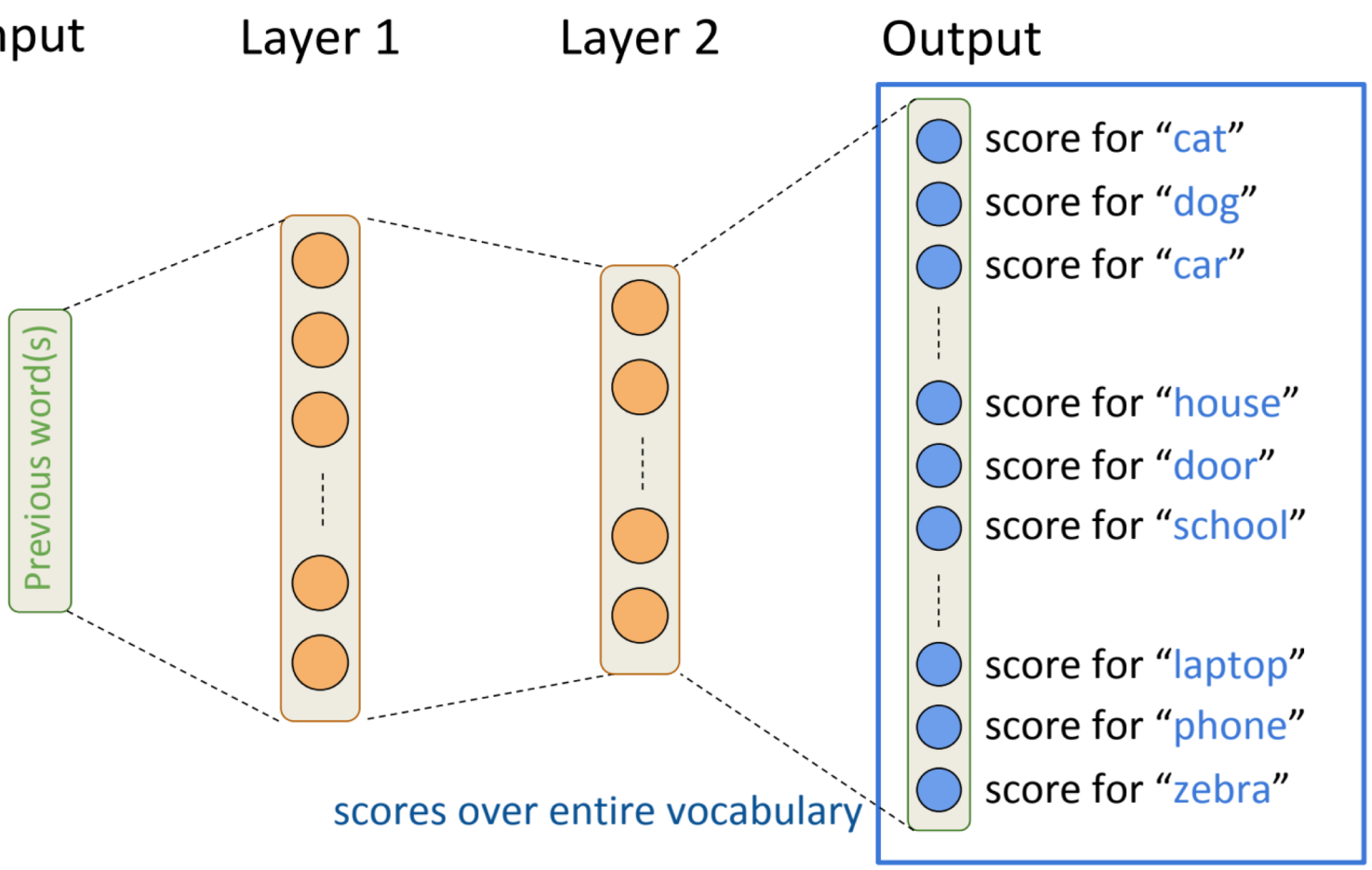
Content

1. What is a Language Model?
2. N-gram Language Model
3. Smoothing
4. Evaluation
5. Management of Large Language Models
6. **Other Approaches**

Continuous Space LMs

- Also referred to as Neural Network LMs
- Words represented as multidimensional vectors (e.g. 300d) in a continuous space
- Neural Network trained to predict a new word (vector) given a sequence of words (vectors)

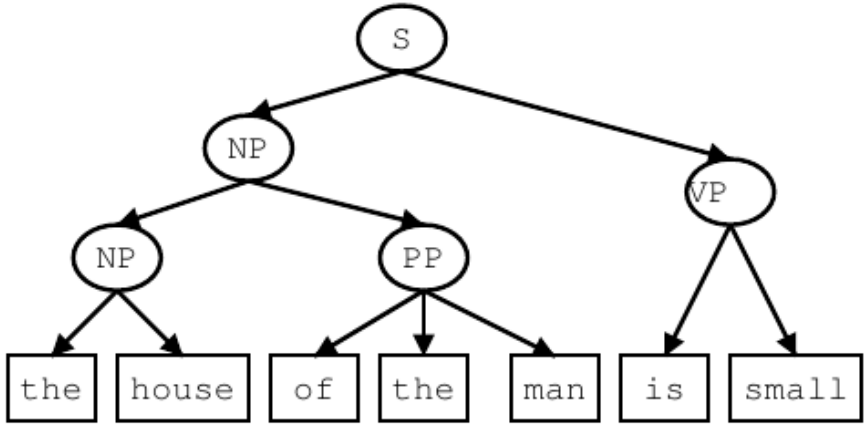
Neural Network Language Model

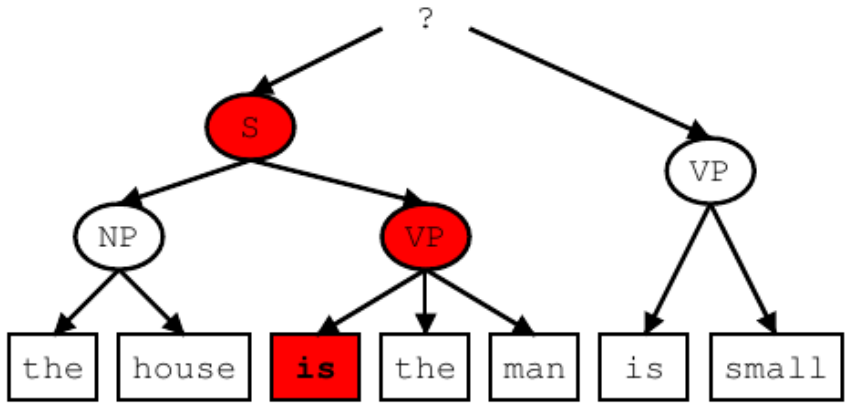


(H. Sajjad)

Syntax Language Models

- Take into account syntactic order and tags
  - p(is|the,house)
  - p(is|VP,house)



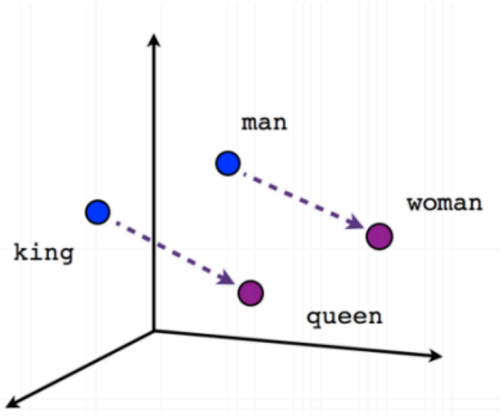


(P. Koehn)

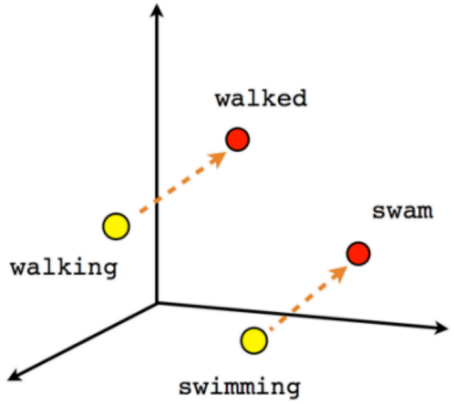
Discussion

Next Week

Dense embeddings



Male-Female



Verb tense

CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computer Science

http://www.cit.ie

Thank you

Haithem. afli@cit.ie

@AfliHaithem