# Lab: NLP & Entity Embedding

**ACTL3143 & ACTL5111 Deep Learning for Actuaries**

## Part A: From text to a design matrix

### Exercise 1

a. Predict the output of each snippet *before* running it.

```
print("Deep\tLearning")
print("for\nActuaries")
print("Sydney\rRome")
```

b. The snippet below was meant to print a Windows file path but doesn't. Explain what went wrong, and give two different one-line fixes.

```
print("C:\tutorials\nlp-lab")
```

c. A claims tokenizer splits on spaces and strips punctuation. For each of the following note fragments, say what is gained or lost, and whether you would want the tokenizer to keep it as one token, split it, or treat it specially:

| Fragment | Question |
|---|---|
| `rear-ended` | one token or two? |
| `not at fault` | does word-by-word keep the meaning? |
| `write-off` | a hyphenated insurance term |
| `$2,500 excess` | what happens to the number and currency? |
| `TPFT` | a domain abbreviation |
| `NSW` | a location that may raise fairness concerns |

Which of these would help predict the *claim type*? Which might create a *privacy or fairness* problem? What is the danger of blindly deleting all punctuation and digits?

## Bag of words

The *bag of words* representation turns each document into a vector of word counts: one column per word in the *vocabulary*, with word order thrown away. In scikit-learn this is `CountVectorizer`, which by default lowercases the text, keeps only tokens of two or more characters, and sorts the vocabulary alphabetically.

This *is* a design matrix. In a GLM your columns might be Age, Exposure, $I(\text{VehBrand} = A)$; here each column is #{occurrences of the word "hail"}, etc. The two are conceptually identical, the only differences are that a text design matrix can have *thousands* of columns, and that the meaning of a column depends heavily on context.

Throughout this section, use this tiny corpus of three claim descriptions:

```
claims = [
    "the policyholder crashed the car",
    "the car was stolen",
    "hail damaged the car roof",
]
```

### Exercise 2

a. In what sense is a bag-of-words matrix "just another design matrix"? Why might it have thousands of columns, and why is that sparsity less alarming than it first looks?

b. Write down the vocabulary that `CountVectorizer` would learn from `claims`, and the $3 \times 9$ count matrix that `fit_transform(claims)` would produce. Check in Python.

c. Using the fitted vectoriser, encode the new claim

> "the policyholder crashed into a stolen truck"

Which words are lost, and why? What does this tell you about *out-of-vocabulary* words?

d. If the vectoriser were instead created with `CountVectorizer(ngram_range=(1, 2))`, how many columns would the matrix for `claims` have? Verify in Python, then give an example of a phrase where bigrams capture meaning a bag of words misses.

e. A real complaint corpus might have a 20,000-word vocabulary. Name two ways to shrink it, and explain the difference between *stemming* and *lemmatisation* (what does each do to "better"?).

## TF-IDF

A flaw of bag of words is that every word is weighted equally, so frequent-but-uninformative words like "the" dominate. *Term frequency–inverse document frequency* re-weights the counts:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times \ln\big(n/\text{df}(t)\big),$$

where $\text{tf}(t, d)$ is the number of times term $t$ appears in document $d$, $n$ is the number of documents, and $\text{df}(t)$ is the number of documents containing term $t$.

### Exercise 3

Use the same `claims` corpus, so $n = 3$.

a. Compute $\text{df}(t)$ for "the", "car", and "crashed".

b. Compute the tf-idf of "the" and "crashed" in the first document.

c. "the" had the *largest* count in the first document yet gets the *smallest possible* tf-idf. Why is that desirable?

d. Compute the tf-idf matrix in Python with `TfidfVectorizer(smooth_idf=False, norm=None)`. The values won't match your hand calculation, read the docs and explain why.

## Word & entity embeddings

Bag-of-words and TF-IDF vectors are *sparse* and *long*, with one interpretable column per word. *Embeddings* instead represent each item, a word or any category, as a *short, dense* vector of learned numbers, where meaning lives in the geometry: similar items sit close together. A *word* embedding and an *entity* embedding are the same idea applied to different "items".

### Exercise 4

A tiny word embedding represents four words by 2-dimensional vectors:

| Word | Vector |
|---|---|
| paris | $(1, 1)$ |
| france | $(3, 1)$ |
| london | $(1, 4)$ |
| england | $(3, 4)$ |

a. Sketch the four points. What do you notice?

b. Using cosine similarity $\text{sim}(\boldsymbol{x}, \boldsymbol{y}) = \dfrac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|}$, compute the similarity of "paris" with "london" and with "france".

c. Compute france − paris + london and find the closest word. What relationship has the embedding captured?

d. GloVe vectors have 300 dimensions. Can you say what dimension 137 *means*? How does that differ from column 137 of a bag-of-words matrix?

e. A famous result (Bolukbasi et al., 2016) had a Word2Vec model complete "man is to computer programmer as woman is to . . . ?" with "homemaker". Given how the arithmetic in (c) works, where does this bias come from, and what does it imply for using pre-trained embeddings in an actuarial model?

f. Place each representation into the right cell: *one-hot encoding*, *bag of words*, *TF-IDF*, *GloVe word embeddings*, *averaged GloVe embeddings*.

| | One **word** → vector | One **document** → vector |
|---|---|---|
| **Sparse** | ? | ? |
| **Dense** | ? | ? |

### Exercise 5

Now the *entity* version of the same idea. An insurer one-hot encodes the policyholder's state (four categories, sorted: NSW, QLD, SA, VIC) and feeds it through a `Dense(2)` layer (no activation) with

$$\boldsymbol{W} = \begin{pmatrix} 0.2 & -0.1 \\ 0.5 & 0.3 \\ -0.4 & 0.0 \\ 0.1 & -0.2 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 0.1 & 0.2 \end{pmatrix}.$$

a. Write the one-hot vector $\boldsymbol{x}$ for a QLD policyholder, and compute $\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}$. How does the result relate to the rows of $\boldsymbol{W}$?

b. Explain why this Dense layer is "just a lookup table", and what an `Embedding` layer changes about the computation and the required input format.

c. Count the parameters in (i) this `Dense(2)` layer and (ii) an `Embedding(input_dim=4, output_dim=2)`. Repeat for 10,000 categories with 10-dimensional embeddings. Is the *parameter* saving the main reason to prefer embeddings?

d. The lecture's rule of thumb sets the embedding dimension to $n^{1/4}$. What does it suggest for the 10,000-category variable? And what is the link back to Exercise 4 — why is a learned `Region` embedding the same machinery as a `word` embedding?

## Part B: A claim-severity classifier

You will analyse a dataset of NSW CTP (compulsory third-party motor) personal-injury claims decided by the Personal Injury Commission. Each case has been summarised into six *clinical* narrative fields (claimant profile, accident mechanism, injuries, treatment, functional impact, and medical evidence), which we have joined into a single `narrative` column for you. Download `ctp_lump_sum_lab.csv` from Moodle and place it beside this notebook; it has that `narrative` column and a `band` target (the lump-sum size: Lower, Middle, or Upper; these correspond to roughly \$1k – \$130k, \$130k – \$320k, and \$320k – \$3.5m), plus `Case Name` and `URL` for reference.

The task: from the narrative, predict the *size band* of the final lump-sum payout (the claim's severity) — the kind of early triage that flags a likely-large claim for a senior assessor or for reserving. Because the narrative describes the *injury and its treatment* rather than the Commission's reasoning and orders, this is a genuine prediction, not a description of a decision already made.

> **!** A note on the data: these are real injury claims. The narratives are LLM-generated summaries of the raw data, which describe physical and mental injuries. If needed, try to avoid reading too many of the raw inputs, or one can consider testing the techniques on a different dataset.

### Exercise 6

a. Load `ctp_lump_sum_lab.csv` with `pd.read_csv`. How many rows are there, how balanced are the bands, and how long is a typical narrative?

b. Create a reproducible, *stratified* train/test split with the `narrative` text as input and the band as target. Why stratify? Why must any vectoriser be fitted on the training set only? When might an insurer prefer a *time-based* split instead?

c. Compute the "do nothing clever" benchmark with a `DummyClassifier`. What does it predict, and what accuracy does it get for three equal bands? Why report it at all?

d. Fit a bag-of-words logistic regression in a `Pipeline` (`CountVectorizer(min_df=5, stop_words="english")` → `LogisticRegression(max_iter=1000, class_weight="balanced")`). Report accuracy and macro-F1. How many features did it build? How is this like fitting a GLM with many indicator columns?

e. Swap counts for TF-IDF and add bigrams (`TfidfVectorizer(min_df=5, max_df=0.8, stop_words="english", ngram_range=(1, 2))`). Did it help? What does `max_df=0.8` remove, and why might bigrams like "soft tissue" or "brain injury" beat the single words?

### Exercise 7

A number on its own is not actuarial work. Now interpret the model and decide whether it is fit for purpose.

a. Plot the confusion matrix for the TF-IDF model. Which bands get confused, and is that *reasonable* (think about where the tercile boundaries fall)?

b. Pull out a few misclassified cases and read them. Do you side with the model or the band label? (This is the NLP version of residual analysis.)

c. For each band, print the terms with the largest positive coefficients. Do they make sense as severity signals? (Because the offer/submission columns were removed, you should see clinical words — injuries, surgeries, work capacity — rather than dollar amounts.)

d. Even with the obvious money columns gone, the narratives are still AI-written summaries of the *full* decision — why is that a subtler leakage risk than an offer figure? Write a two-or-three sentence judgement note: would you use this model to triage large claims, to set reserves, to price, or not at all, and why?

## Part C: a neural embedding model (optional)

This last exercise is a stretch goal, which is to build our own word embeddings and average them into sentence/document embeddings.

Note, the performance with just ~500 rows will be rather poor. Also, the `GlobalAveragePooling1D` will be taught in the Computer Vision lecture.

### Exercise 8

a. Turn each narrative into a sequence of integer word-indices. (We reuse a `CountVectorizer` just to *build the vocabulary and tokenizer*, reserve index 0 for padding, then `pad_sequences` to a fixed length.)

b. Build a small Keras model: `Embedding` → `GlobalAveragePooling1D` → `Dense` → softmax over the three bands. What does the `Embedding` layer learn here, and why is averaging the word vectors a reasonable (if crude) document representation?

c. Did the neural model beat TF-IDF + logistic regression? Why might the *simpler* model win on a dataset this small?

d. In one or two sentences: how is the `Embedding` layer in this model the same object as the `Region`/`VehBrand` embedding from the lecture? What is the only thing that differs?

Hint: you may benefit from `keras.utils.pad_sequences`.

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. *Advances in Neural Information Processing Systems*, *29*.