LLMs for Social Scientists

POL42560 AI and Large Language Models

Lorcan McLaren

February 2026

Table of contents

Welcome

This is the course website for **POL42560 AI and Large Language Models**, taught at University College Dublin in the Spring Trimester 2026.

Overview

Large language models (LLMs), such as those behind tools like ChatGPT, have garnered significant attention for their ability to generate human-like text, sparking both enthusiasm and debate about their implications for various fields. For social scientists, LLMs offer a potentially useful approach to processing and interpreting vast amounts of data, enhancing our ability to study complex societal issues.

This course offers an interdisciplinary approach to understanding and applying LLMs in social and political science, with a focus on text analysis. It combines theoretical foundations with practical, hands-on experience in applying LLMs to address substantive social science questions. Students will explore the capabilities and limitations of LLMs and engage critically with issues such as bias, environmental impact, misinformation, and intellectual property rights. They will also become familiar with essential research practices including documentation, reproducibility, and validation.

Learning Outcomes

By the end of the course, students will be able to:

1. Understand the fundamentals of large language models and their applications in social sciences.
2. Implement LLMs for various tasks relevant to political and social science research using Python.
3. Critically evaluate the ethical and societal implications of AI technologies, particularly LLMs, in the context of social science.
4. Apply best practices in research documentation, reproducibility, and validation when using AI tools.
5. Engage in informed discussions about the impact of AI on society, including issues of bias, sustainability, and concentration of power.

Prerequisites

This course is designed for students with or without experience in Python, but a willingness to learn quickly and engage with technical content is essential. Students without prior programming experience would benefit from taking **POL42340 Programming for Social Scientists** alongside this course to familiarise themselves with Python fundamentals.

POL42050 Quantitative Text Analysis provides a broader overview of text-as-data approaches, complementary to the content of this course.

Course Details

Time	Fri, 11:00 – 12:50
Format	In-person lectures and labs
Credits	10, Level 4
Instructor	Lorcan McLaren
Email	lorcan.mclaren@ucdconnect.ie
Office Hours	Wed, 15:00 – 16:00 (sign up)

Part I

Course Information

Syllabus

Course Structure

Each class involves a lecture introducing the topic, followed by a hands-on lab session where we work on exercises related to the lecture content. Students are expected to complete the readings before class each week, and to submit their lab work via Brightspace.

See the Weekly Content chapters for detailed content for each session.

Schedule at a Glance

Week	Topic	Lab
1	Introduction	Setting up a Python development environment
2	Text-as-Data	Dealing with textual data in Python
3	Embeddings	Measuring text similarity using embeddings
4	Transformers	Fine tuning transformers for text classification
5	Validation and Performance Measurement	Measuring model performance with `scikit-learn`
6	Generative Language Models I	Introduction to HuggingFace `transformers`
7	Generative Language Models II	Introduction to `langchain`
–	*Study Break*	

Week	Topic	Lab
8	Interpretability, Explainability and Bias	Applying interpretability tools; visualising bias
–	*Good Friday*	
10	Ethical Use of LLMs?	Measuring environmental costs of LLM use
11	Beyond Text	Working with images, audio and video in Python
12	Project Presentations	

Textbooks and Readings

All mandatory readings are indicated in the weekly chapters and are open access or available through the UCD Library. Readings should be completed ahead of class each week.

Approachable

- Alammar, J. and Grootendorst, M. (2024) *Hands-On Large Language Models: Language Understanding and Generation.* 1st edition. O'Reilly Media.
- Grimmer, J., Roberts, M.E. and Stewart, B.M. (2022) *Text as Data: A New Framework for Machine Learning and the Social Sciences.* Princeton University Press.

Deeper Dive

- Raschka, S. (2024) *Build a Large Language Model (From Scratch)*. Manning.
- Kamath, U. et al. (2024) *Large Language Models: A Deep Dive: Bridging Theory and Practice.* Springer.

Assessment

Components

Component	Deadline	Grade %
Lab participation	Continuous	20
Group presentations	April 24	20
Group project	May 1	60

Lab Participation

Each lab session includes a Jupyter notebook that must be submitted through Brightspace for participation credit. While your solutions don't need to be perfect, they should demonstrate a genuine attempt at completing all exercises. Include clear comments throughout your code to show your understanding of the concepts.

You should **execute all code cells prior to saving**, so that the output for each cell is visible in the submitted version.

You will have **one week to submit the notebook** from each lab session, up to the start of the subsequent class (e.g., the notebook for Week 2 must be submitted by 08:59 on Friday in Week 3). You may miss up to one submission without prior approval. Late submissions will not be accepted unless an extension has been granted, as exercise solutions may be discussed in class.

Group Presentation

Each group must deliver a presentation lasting **no more than 12 minutes**, using either LaTeX Beamer, PowerPoint or Google Slides. Submit your slides on Brightspace by **23:59 on Wednesday, April 22**. Groups should divide the presentation between 2–3 speakers, with non-presenting members taking responsibility for answering questions from peers and the instructor during the Q&A session.

You should prepare together by practising the presentation and anticipating potential questions. The same grade will be awarded to all group members, unless one or more members fail to participate in both the presentation and Q&A.

Your presentation should cover:

1. Research question and importance
2. Theoretical expectations and main hypothesis
3. Data description
4. Methodology for hypothesis testing
5. Initial results (if available)
6. Challenges encountered and potential risks for next steps (if applicable)

Group Project

Each group must submit a **4,000–5,000 word** research paper (**excluding references and appendices**) as well as the code used to conduct the analysis. All students in a group will be awarded the same grade, but you may be asked to indicate who contributed to which parts of the project.

You have free choice in selecting your research question, but you must justify why it is relevant and important for understanding or addressing a meaningful social science question. You may either collect original data appropriate for your research question or identify and use existing datasets (see Appendix A), clearly explaining your data source selection. Your analysis must incorporate **at least one** of the methods covered in this course – embeddings, transformers, or generative models – in a way that meaningfully advances your research objectives.

Submission Format

1. A research paper in PDF format. The paper must be written in LaTeX using the collaborative Overleaf document that will be created for each group.
2. A GitHub repository containing all code needed to reproduce your analysis.

Research Paper Structure

The paper should contain the following sections:

1. **Introduction** – Research question/puzzle, significance, main hypothesis
2. **Theory** – Theoretical framework, prior findings on DV-IV relationship, expected effects
3. **Data & Methods** – Dataset overview, unit of analysis, sample size & missing data, analysis approach

4. **Results** – Empirical findings, statistical tests, key patterns
5. **Conclusion** – Link to research question, main insights, limitations, broader implications

Software

All code for this course should be written in **Python**. For those who haven't worked with this language previously, POL42340 Programming for Social Scientists runs in parallel to this course and provides a comprehensive introduction to Python.

Development Environment

We will use **VS Code** to write and execute code locally. For some lab sessions and the group project, where your personal device may not be powerful enough or where long execution times are expected, **Google Colab** should be used. Colab provides limited free access to T4 GPUs, which will enable you to run code that cannot be executed locally without a GPU.

Package Management

We will use `venv` and `pip` to manage our Python environment and install packages (though feel free to use Anaconda if you are more familiar with this platform). I strongly encourage the use of **Git** for version control, particularly when working on your group project; however, we will not cover this extensively in class.

Useful Python Packages

The following packages are particularly useful for dealing with textual data:

Package	Description
`spaCy`	Industrial-strength NLP library
`NLTK`	Classic NLP toolkit
`Gensim`	Topic modelling and word embeddings
`sentence-transformers`	Sentence and document embeddings
`transformers`	HuggingFace model hub and pipelines
`langchain`	Framework for building LLM applications
`scikit-learn`	Machine learning and evaluation metrics

Policies

AI Use

I encourage the use of generative AI tools when completing the assignments for this module, but all work relying on AI-generated content must adhere to the highest academic standards. Users of this technology must be aware of what it can and, more importantly, what it cannot do well. It is crucial for you to exercise judgement when evaluating the quality and reliability of content generated through AI platforms.

AI is not a panacea for all writing challenges; it will not automatically generate a flawless, logically coherent, and factually correct assignment. Instead, use AI as a tool to tackle specific issues such as brainstorming and idea formation, literature discovery, and text drafting. View your preferred AI platform(s) as useful but imperfect tools that can offer inspiration, new perspectives, and supplementary areas for research for your own work. In-depth research on your part remains essential to ensure coherent, factual, and scientifically informed perspectives in your assignment. Always cross-reference the information AI offers against other independent and reliable sources.

AI use must be in line with UCD's policies on academic integrity.

Documenting AI Use

You are expected to provide an account of the tools used and how they were used in a **mandatory appendix** to your group project. This appendix will be assessed as part of each assignment, with grade points awarded for effective communication of the methods used to generate content.

For each instance where a generative AI tool is used, you need to provide:

1. An in-text citation or footnote
2. A bibliographic reference to the tool used and the date of access
3. An entry in the mandatory AI appendix detailing how the tool was used

For code, while I do not expect you to keep a log of all prompts used, I do expect you to add comments indicating where code has fully or partially been generated by a tool such as ChatGPT or GitHub Copilot.

Suggested AI Tools

- **ChatGPT** – good at writing code
- **Claude** – better than ChatGPT at writing text
- **Gemini** – fast and useful integrations
- **Perplexity** – great for finding relevant web pages and documents
- **HuggingChat** – open source, flexible, and free

Plagiarism

Academic integrity is one of the core values of the UCD Education Strategy. Plagiarism is the inclusion, in any form of assessment, of material without due acknowledgement of its original source. See UCD's Academic Integrity Policy for full details.

Late Submission

Lab notebooks submitted late will not be accepted unless an extension has been granted. For group projects submitted past the due date:

- Coursework received within **5 working days** after the due date will have the grade reduced by **one grade point**.
- Coursework received between **5 and 10 working days** after the due date will have the grade reduced by **two grade points**.
- Coursework received **more than 10 working days** after the due date will not be accepted.

Office Hours

Weekly office hours during the teaching term: Wednesdays, 15:00–16:00. Book a slot via Calendly if you wish to attend, either in person or online.

Syllabus Modification Rights

I reserve the right to reasonably alter the elements of the syllabus at any time to keep pace with the course schedule. If adjustments are made, the revised syllabus will be uploaded to Brightspace.

Part II

Weekly Content

1 Introduction

ℹ Week 1

Lecture: Module overview and organisation
Lab: Setting up a Python development environment

1.1 Lecture

1.1.1 Module overview

This module takes a utilitarian approach to AI and large language models (LLMs): how can we maximise the utility of LLMs and generative AI for us as political and social scientists, while minimising the harms? The module is research-led and driven by readings from current scholarship, with an emphasis on active and collaborative learning and transparent and reproducible practices.

Companion modules

POL42340 Programming for Social Scientists runs in parallel and provides an introduction to Python that is highly useful if you do not have prior Python experience. **POL42050 Quantitative Text Analysis** provides a broader overview of text-as-data approaches that is complementary to this course.

The primary focus of the module is on LLMs and their applications in social science research, particularly using text-as-data. We will not focus extensively on other AI technologies such as speech-to-text, text-to-speech, or music/image/audio generation, as these are less practically relevant to social science research at present. Image recognition and multimodality may be touched upon, but these remain new tools for social science and are not yet in widespread use in scholarship. The module also considers the implications of these technologies for politics and society.

1.1.2 A brief overview of AI

1.1.2.1 What is artificial intelligence?

Artificial intelligence can be defined as a rational agent: an automated system that does the "right thing", based on its objective, given its environment (**russell2020artificial?**). In the past, attempts at creating AI were dominated by a rules-based approach (or "expert systems"), in which knowledge and logic are encoded as a set of rules and a system makes decisions based on these. More recently, machine learning approaches have come to predominate. While machine learning and neural networks have existed since the 1960s and saw key innovations in the 1980s, it was increases in computational power and the rise of big data in the 2000s that led to a shift away from rules-based systems. Machine learning systems are able to handle uncertainty by learning patterns from a large training dataset in a way that often surpasses the performance of more brittle, handwritten rules.

AI, machine learning, and deep learning are nested concepts. AI is the broadest category, machine learning is a subset of AI, and deep learning is a subset of machine learning.

1.1.2.2 Types of machine learning

Machine learning can be divided into three broad categories:

- **Supervised learning** is when we have a training dataset for which we know the outcomes or "labels", and use this to train a model for unseen data. *Classification* is when we wish to train a model to place observations into discrete categories (e.g. categorising images as containing cats or dogs). *Regression* models have continuous rather than discrete outcomes (e.g. predicting property prices).
- **Unsupervised learning** is when the outcomes or "labels" are not known for our dataset. For example, we might group similar images into clusters, without explicitly knowing how these clusters are defined. We might later find that these clusters correspond to discrete categories (e.g. animals), but we do not know this beforehand.
- **Reinforcement learning** is where a model is trained to make a series of decisions sequentially, such that each outcome is dependent on the previous, rather than being independent as in supervised learning. The goal is to find the optimal set of actions that maximise expected rewards (e.g. AI models that learn to play games).

1.1.2.3 Neural networks and deep learning

One well-known family of machine learning architecture is the neural network. These are networks of nodes (or "neurons") connected to each other by weighted edges. Each node is an individual mathematical function. Input data passes through layers to generate predictions or classifications. Neural networks can be used for both supervised and unsupervised learning.

Deep learning refers to the use of "deep" neural networks with one or more hidden layers. Each layer extracts increasingly complex features from the data. Deep neural networks learn useful feature representations automatically, allowing them to handle various types of data from text to image to audio.

1.1.3 Large language models

LLMs are fundamentally prediction machines — they predict the next token (word or part of word) based on the previous context. Through this simple objective and massive training data, they appear to learn complex language understanding. Key characteristics include:

- Training on internet-scale datasets
- The ability to handle multiple tasks without task-specific training
- Learning patterns of language and knowledge during pre-training

The hype around LLMs stems from their apparent ability to perform tasks previously thought to require human intelligence, including the emergence of "reasoning" capabilities through chain-of-thought prompting, the ability to follow instructions and adapt to new tasks, and integration into research workflows. However, it is important to maintain perspective: despite the module's name, we should avoid the use of "artificial intelligence" in favour of more specific terminology. LLMs are fundamentally statistical pattern matching systems, with significant limitations. They do not "think" or "reason" — they predict patterns. They can be confidently wrong and require careful human oversight.

1.1.4 Why should political (or social) scientists care?

Political scientists need to engage with LLMs in order to understand:

1. **Policy implications and governance challenges** posed by these technologies
2. **How to use these tools responsibly in research**
3. **Their impact on information environments**

1.2 Lab

Setting up a Python development environment.

1.3 Readings

- Grossmann, I. et al. (2023) ‘AI and the transformation of social science research’, *Science*, 380(6650), pp. 1108–1109. https://doi.org/10.1126/science.adi1778
- Ziems, C. et al. (2023) ‘Can Large Language Models Transform Computational Social Science?’ *arXiv*. http://arxiv.org/abs/2305.03514

2 Text-as-Data

ℹ Week 2

Lecture: Tasks in text-as-data (scaling, topic modelling, classification, clustering, keyword extraction, NER, sentiment analysis)
Lab: Dealing with textual data in Python

2.1 Lecture

Text is one of the main ways we study politics. Most features of politics that we might be interested in have been or can be encoded as textual data in some way: opinion, motivation, actions, framing, outcomes. Opinion polling and experiments are distinct, but almost all other research approaches, whether qualitative or quantitative, attempt to analyse text in one form or another.

2.1.1 Content analysis

Content analysis is used to understand the (self-)presentation of political actors and policy content. It uses human judgement not to make sense of each text directly, but instead to apply a scheme to convert the text into data by recording labels or ratings for each unit of text. Text reflects how politicians seek to project an image of themselves, their actions, and others. For example, we might review a series of speeches from MPs on budgetary measures and annotate or "code" them as fiscally right-wing or left-wing. Content analysis can also be applied to legal texts and policy documents.

2.1.1.1 Human vs. automated content analysis

Humans are flexible and able to handle nuance and ambiguity, but are vulnerable to subjectivity and performance variability over time. Even trained expert coders — the "gold standard" — are not failure-proof. The Manifesto Project, a major dataset of political texts annotated at the sentence-level with labels for policy area, has been found to have levels of inter-rater agreement and reliability so low that, had the coders been oncologists, their levels of tumour misdiagnosis would have been medically and financially catastrophic (**mikhaylov2012coder?**).

Automated methods are more consistent, cost- and time-efficient, but they may be consistently wrong. Humans are still typically used as the point of reference ("ground truth") against which we compare other approaches.

2.1.2 Quantitative text analysis

In quantitative text analysis, we convert each document into some kind of numeric representation which we can work with more easily. The document-feature matrix (DFM) is one widely used representation that records the frequency of each feature (typically a word) across each document. From even a simple DFM, we can make initial observations — for instance, comparing how frequently different US presidents mention terms like "economy", "crime", or "climate" in their State of the Union addresses.

Four key principles guide quantitative text analysis (**grimmer2013text?**):

1. All quantitative models of language are wrong — but some are useful.
2. Quantitative methods for text amplify resources and augment humans.
3. There is no globally best method for automated text analysis.
4. Validate, validate, validate.

2.1.3 Process

The typical text-as-data workflow follows seven steps (**benoit2020text?**):

1. **Selecting texts and defining the corpus.** Identify the corpus of texts relevant to the research question. Texts are generally distinguished from one another by attributes relating to the author, speaker, time, or topic. Be aware of selection issues and data gaps (e.g. coalition manifestos tend to disappear after a coalition collapses, leading to a bias in the dataset towards more stable coalitions).

2. **Converting texts into a common electronic format.** This involves converting PDFs and image formats to text via OCR, standardising file types (HTML, XML, JSON, Word, Excel, TXT, TSV, CSV), handling character encoding issues (e.g. accents or special characters), and text cleaning (whitespace, page numbers). This step is often glossed over, but getting or creating a clean, organised dataset can be one of the most time-consuming components of any text-as-data project.

3. **Defining documents and choosing the unit of analysis.** Documents could be individual social media posts, newspaper articles, parliamentary speeches, and so on. Consider what level of granularity is needed to answer the research question. Texts are often analysed at the sentence level and then aggregated afterwards.

4. **Defining and refining features.** A "type" refers to the abstract category of a word or concept, while a "token" is a specific instance of that type as it appears in text. Preprocessing steps include tokenisation, casefolding, stemming, lemmatisation, stopword removal, and the identification of multi-word expressions (n-grams). Tokenisation can be more challenging in languages such as Chinese (with no spaces between words) or in morphologically rich languages like Finnish or Turkish.

5. **Converting textual features into a quantitative matrix.** This yields an N×M matrix, where N is the number of documents and M is the number of features. Values typically start as raw frequency counts but may be re-weighted via normalisation or tf-idf (a measure of importance of a word to a document in a collection, adjusted for the fact that some words appear more frequently in general). Trimming may be applied to overcome sparsity by filtering based on term frequency.

6. **Analysing the matrix data using an appropriate statistical procedure.**

7. **Interpreting and reporting the results.**

2.1.4 Applications

2.1.4.1 Characterising text

Keyword extraction and analysis involves identifying the most frequent or distinctive terms in a corpus. For example, keyword datasets have been used to track how European political parties on Facebook discuss the migration crisis, with variation across countries and ideological groups (**caravaca2025european?**).

Named entity recognition (NER) identifies people, organisations, and other entities mentioned in text. (**jaros2018china?**) used NER in Chinese official provincial newspapers to recognise all people and organisations named in Party newspapers, using the frequency of mentions as a measure of power. They found that Xi Jinping was much more frequently mentioned in Party newspapers than his predecessor.

Sentiment analysis measures the emotional tone of text. (**rheault2016measuring?**) tracked the emotional polarity of government and opposition speeches in Britain from 1946 to 2013.

Readability analysis assesses how easy texts are to understand. (**benoit2019measuring?**) examined the readability of US State of the Union addresses over time, finding a trend towards simpler language.

2.1.4.2 Classifying into known categories

Dictionary analysis uses predefined lists of words to classify text. Dictionaries are interpretable but brittle: they do not generalise well to other datasets and are vulnerable to polysemy (e.g. the word "kind" can mean both "type" and "generous").

Supervised machine learning is less interpretable but more flexible. A model is trained on labelled examples and then applied to classify unseen documents. For example, (**muller2024nostalgia?**) used DistilBERT to measure party-level nostalgia across European democracies.

2.1.4.3 Discovering categories

Topic modelling is an unsupervised method in which clusters of co-occurring words represent "topics" and documents contain them in relative proportion. Estimated topics are unlabelled, so a human must assign labels by interpreting the content of the words most highly associated with each topic. Choosing the correct number of topics involves balancing statistical measures and interpretability.

2.1.4.4 Measuring latent features

Supervised scaling methods such as Wordscores (**laver2003extracting?**) require some reference texts for which a score (e.g. left–right position) is known, from which the model learns to score new texts.

Unsupervised scaling methods such as Wordfish (**slapin2008scaling?**) run without reference texts. It is up to the researcher to interpret what the estimated dimensions actually represent. A risk is that the model might capture thematic differences (e.g. talking about the economy vs. foreign policy) rather than ideological differences.

> ℹ Note
>
> The techniques discussed this week rely on document-feature matrices as the numeric representation of text. Next week, we will introduce another representation — embeddings — which underpin many modern text analysis methods and are fundamental to large language models.

2.2 Lab

Text preprocessing and document-feature matrices in Python.

2.3 Readings

- Grimmer, J. and Stewart, B.M. (2013) ‘Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts’, *Political Analysis*, 21(3), pp. 267–297. https://doi.org/10.1093/pan/mps028

3 Embeddings

Week 3

Lecture: Word embeddings, sequence embeddings, and document embeddings
Lab: Measuring text similarity using embeddings

3.1 Lecture

Last week, we discussed document-feature matrices as one way of representing words and documents numerically. This week, we introduce another approach — embeddings — which are used for many modern text analysis methods and are also fundamental to large language models.

3.1.1 From bag-of-words to embeddings

3.1.1.1 The bag-of-words model

The bag-of-words (BoW) model is one of the simplest ways to represent text numerically. A sentence such as "The cat sat on the mat" would be represented as a vector of word counts: *the*: 2, *cat*: 1, *sat*: 1, *on*: 1, *mat*: 1. This approach has several important limitations: it loses word order ("the cat sat" and "sat the cat" would be identical), it cannot capture semantic relationships between words, and it creates very sparse, high-dimensional vectors since most words in a vocabulary will not appear in any given document.

3.1.1.2 Distributional semantics

Embeddings are grounded in the principle of distributional semantics, often summarised by the phrase: "you shall know a word by the company it keeps." The idea is that meaning is learned and defined by context — we learn to associate a particular word with a particular object or referent by encountering it repeatedly in different contexts.

3.1.2 What are embeddings?

Embeddings represent a fundamental shift from counting words to understanding their meaning through context. Unlike bag-of-words representations, embeddings are:

- **Dense vectors** (most values are non-zero)
- **Semantic** (similar words have similar vectors)
- **Compact** (typically 100–300 dimensions, compared to vocabulary-sized vectors)

Each word is mapped to a fixed-length vector. Words that appear in similar contexts have similar vector representations, and the distance between vectors corresponds to semantic distance (measured by Euclidean distance, cosine similarity, Manhattan distance, or other metrics).

3.1.2.1 Understanding vector dimensions

A 2-dimensional embedding is like coordinates on an X/Y axis. In 3D, we add a Z axis. Beyond three dimensions, it becomes difficult to visualise the space, but higher-dimensional representations are mathematically useful and capture richer semantic relationships. There are methods (such as PCA or t-SNE) that allow us to reduce this space back down to 2D or 3D for visualisation and interpretation.

3.1.3 Geometric properties of embeddings

Visualising embeddings reveals patterns in how words relate to each other. Semantically similar words cluster together: “cat” and “kitten” have high similarity (near synonyms), “cat” and “dog” have moderate similarity (same category), while “cat” and “house” have low similarity (unrelated concepts). This mirrors human intuition about word relationships.

3.1.3.1 Vector arithmetic

One striking property of embeddings is that vector arithmetic can be used to illustrate semantic relationships. The famous example is:

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

This shows that embeddings capture gender relationships as a consistent directional offset. Other examples include: Paris − France + Italy Rome, and terrible − bad + good great.

These patterns hold even across languages: when words from different languages are embedded in the same space, semantically similar words cluster together regardless of language.

3.1.4 Training word embeddings

Word embeddings can be trained using prediction-based or count-based methods.

Prediction-based models include:

- **Word2Vec** (Google, 2013), which has two architectures: *Skip-gram*, which predicts context words from a target word (e.g. given "cat", predict "The ___ sat"), and *CBOW* (Continuous Bag of Words), which predicts a target word from its context (e.g. given "The ___ sat", predict "cat"). Skip-gram works better for rare words, while CBOW is faster and better for frequent words. The training objective is to maximise the probability of actual context words while minimising the probability of unrelated words.
- **GloVe**, which combines the strengths of count-based and prediction-based methods.
- **FastText**, which handles out-of-vocabulary words by using subword information.

Count-based models create large matrices counting how often words co-occur, then use singular value decomposition (SVD) to reduce dimensionality while preserving important relationships. These are computationally expensive but theoretically well-understood.

In both cases, the embedding vectors are gradually adjusted during training (via backpropagation, similar to neural network training) to minimise prediction errors.

3.1.5 Bias in word embeddings

Word embeddings do not create bias, but they inherit it from the training data. For example, early embeddings trained on news corpora placed "doctor" closer to "man" and "nurse" closer to "woman" in the embedding space.

Debiasing methods have been proposed, but these can result in other kinds of distortion and will inevitably attend to some types of bias more than others. Training data in general disproportionately reflects the WEIRD demographic (Western, Educated, Industrialised, Rich, Democratic).

As social scientists, we generally do not apply debiasing methods because those differences in how social groups are discussed are often interesting to us and are frequently the object of study. However, we must be conscious of these biases and how they impact our results. We will discuss bias more extensively in Week 8.

3.1.6 Applications of embeddings

Embeddings serve two main purposes in research:

1. **As feature representations for machine learning tasks** such as text classification, sentiment analysis, and machine translation.

2. **As objects of direct study**, enabling researchers to analyse how language and meaning vary across groups, contexts, and time.

3.1.6.1 Case study: Gender and party differences in the US House

(**rodriguez2023embedding?**) used word embeddings trained separately on speeches by different groups of US legislators to examine how the same words are used with different meanings or connotations. While male and female legislators, and Republicans and Democrats, tend to use common words like "also" and "but" in similar ways, they diverge when it comes to more politically charged terms. In particular, male legislators use "marriage" in very different ways to female legislators, while Republicans refer to "immigration" in very different contexts to Democrats. Looking at nearest-neighbour terms in the embedding space, Democrats tend to use terms related to "reform" when discussing immigration, while Republicans tend to focus on "enforcement."

3.1.6.2 Case study: Gender in school textbooks

(**lucy2020content?**) examined how different populations are discussed in K–12 textbooks in Texas using word embeddings. They found that words related to power and achievement are more associated with terms referring to men, while words related to home life and labour are more associated with women (the labour association likely reflecting historical discussion of women entering the workplace).

3.1.7 Sentence, paragraph, and document embeddings

While word embeddings represent individual words, we can also compute embeddings for longer text sequences — sentences, paragraphs, or entire documents. These are used for clustering, topic modelling, classification, legislative text comparison, policy area classification, speaker comparison, and ideological scaling.

3.1.7.1 Case study: Party positions in UK Parliament

(**rheault2020word?**) calculated party embeddings from parliamentary debates and applied PCA. The first principal component (x-axis) captured the left–right divide, while the second (y-axis) captured the government–opposition divide. The resulting plot shows how parties shifted over time, with the Thatcher years being furthest to the right and Labour moving rightward during the Blair years under New Labour. The authors also calculated embeddings for individual legislators, allowing comparisons between individual MPs rather than just between parties.

3.1.8 Embeddings beyond text

Embedding techniques are not limited to text. Audio embeddings can be used for speaker verification and song recommendations. CLIP embeddings allow text and images to be embedded in the same vector space, so that images and their descriptions appear close together in the space. Multimodal models build on this principle to process and generate across different data types.

3.2 Lab

Working with word embeddings in Python.

3.3 Readings

- Rheault, L. and Cochrane, C. (2020) 'Word Embeddings for the Analysis of Ideological Placement in Parliamentary Corpora', *Political Analysis*, 28(1), pp. 112–133. https://doi.org/10.1017/pan.2019.26
- Rodriguez, P.L. and Spirling, A. (2022) 'Word Embeddings: What Works, What Doesn't, and How to Tell the Difference for Applied Research', *Journal of Politics*, 84(1), pp. 101–115. https://doi.org/10.1086/715162

4 Transformers

ℹ Week 4

Lecture: Basics of transformer models
Lab: Fine tuning transformers for text classification

4.1 Lecture

Content to be added.

4.2 Lab

Content to be added.

4.3 Readings

- Widmann, T. and Wich, M. (2023) ‘Creating and Comparing Dictionary, Word Embedding, and Transformer-Based Models to Measure Discrete Emotions in German Political Text’, *Political Analysis*, 31(4), pp. 626–641. https://doi.org/10.1017/pan.2022.15
- Laurer, M. et al. (2024) ‘Less Annotating, More Classifying: Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT-NLI’, *Political Analysis*, 32(1), pp. 84–100. https://doi.org/10.1017/pan.2023.20

5 Validation and Performance Measurement

ℹ Week 5

Lecture: Performance metrics, human coding, and cross-validation
Lab: Measuring model performance with `scikit-learn`

5.1 Lecture

Content to be added.

5.2 Lab

Content to be added.

5.3 Readings

- Birkenmaier, L., Lechner, C.M. and Wagner, C. (2024) 'The Search for Solid Ground in Text as Data: A Systematic Review of Validation Practices and Practical Recommendations for Validation', *Communication Methods and Measures*, 18(3), pp. 249–277. https://doi.org/10.1080/19312458.2023.2285765

6 Generative Language Models I

ℹ Week 6

Lecture: Basics of decoder-only models
Lab: Introduction to HuggingFace `transformers`

6.1 Lecture

Content to be added.

6.2 Lab

Content to be added.

6.3 Readings

- Heseltine, M. and Hohenberg, B.C. von (2024) 'Large language models as a substitute for human experts in annotating political text', *Research & Politics*. https://doi.org/10.1177/20531680241236239
- Gilardi, F., Alizadeh, M. and Kubli, M. (2023) 'ChatGPT outperforms crowd workers for text-annotation tasks', *Proceedings of the National Academy of Sciences*, 120(30), p. e2305016120. https://doi.org/10.1073/pnas.2305016120

7 Generative Language Models II

> Week 7
>
> **Lecture:** Advanced prompting
> **Lab:** Introduction to `langchain`

7.1 Lecture

Content to be added.

7.2 Lab

Content to be added.

7.3 Readings

- Abdurahman, S. et al. (2024) 'Perils and opportunities in using large language models in psychological research', *PNAS Nexus*, 3(7), p. 245. https://doi.org/10.1093/pnasnexus/pgae245
- Barrie, C., Palmer, A. and Spirling, A. (2024) 'Replication for Language Models'. https://arthurspirling.org/documents/BarriePalmerSpirling_TrustMeBro.pdf

8 Interpretability, Explainability and Bias

ℹ Week 8

Lecture: Techniques for interpretability in text classification; sources of bias and implications
Lab: Applying `transformers-interpret` and `ferret` for interpreting model predictions; visualising bias

8.1 Lecture

Content to be added.

8.2 Lab

Content to be added.

8.3 Readings

- Wan, Y. et al. (2023) '"Kelly is a Warm Person, Joseph is a Role Model": Gender Biases in LLM-Generated Reference Letters'. *arXiv*. https://doi.org/10.48550/arXiv.2310.09219
- Rossi, L., Harrison, K. and Shklovski, I. (2024) 'The Problems of LLM-generated Data in Social Science Research', *Sociologica*, 18(2), pp. 145–168. https://doi.org/10.6092/issn.1971-8853/19576

9 Ethical Use of LLMs?

ℹ Week 10

Lecture: Ethical concerns in developing and deploying LLMs
Lab: Measuring environmental costs of LLM use

9.1 Lecture

Content to be added.

9.2 Lab

Content to be added.

9.3 Readings

- Williams, A., Miceli, M. and Gebru, T. (2022) 'The Exploited Labor Behind Artificial Intelligence', *Noema*, 13 October.
- Bender, E.M. et al. (2021) 'On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?', in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. New York, NY, USA: Association for Computing Machinery (FAccT '21), pp. 610–623. https://doi.org/10.1145/3442188.3445922
- Bucher, M.J.J. and Martini, M. (2024) 'Fine-Tuned "Small" LLMs (Still) Significantly Outperform Zero-Shot Generative AI Models in Text Classification'. *arXiv*. https://doi.org/10.48550/arXiv.2406.08660

10 Beyond Text

Week 11

Lecture: Social science applications for multimodal data
Lab: Working with images, audio and video in Python

10.1 Lecture

Content to be added.

10.2 Lab

Content to be added.

10.3 Readings

- Luken, M. et al. (2024) 'MEXCA - A Simple and Robust Pipeline for Capturing Emotion Expressions in Faces, Vocalization, and Speech'. *OSF*. https://doi.org/10.31234/osf.io/56svb

11 Project Presentations

ℹ Week 12

Group presentations

This session is dedicated to group presentations. See the Assessment chapter for presentation requirements and grading criteria.

A Datasets

Explore the following datasets for use in your group project. These datasets vary in format and accessibility – some are readily available in CSV format while others may require additional wrangling or scraping. You are also welcome to find other datasets to work with or to scrape your own data from news sites or other sources. While extra effort in data collection will be considered in grading, remember that the main focus is applying the methods covered in this module.

A.1 Party Manifestos and Political Programs

- Manifesto Project
- The American Presidency Project

A.2 Parliamentary Speeches

- **ParlEE** (European Parliament speeches)
- UN General Debate Corpus
- US Congressional Record
- Irish Oireachtas Record

A.3 Political Advertising

- Meta Ads Library API

A.4 News and Legal Documents

- LexisNexis (available through UCD Library)

A.5 Collections

The following links collect multiple datasets for you to explore. Some of these are textual data, while others may be useful to augment your analysis (e.g., data on party ideology, vote share, government/opposition status and more).

- Irish Politics Data
- PolData
- NLP Datasets
- Google Dataset Search – Political Science

References