NBER WORKING PAPER SERIES

# DEEP LEARNING FOR ECONOMISTS

Melissa Dell

Working Paper 32768
http://www.nber.org/papers/w32768

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
August 2024

I would like to thank four anonymous referees, the editor, and Jake Carlson for their helpful comments and suggestions. Yiyang Chen provided excellent research assistance. The views expressed herein are those of the author and do not necessarily reflect the views of the National Bureau of Economic Research.

Deep Learning for Economists
Melissa Dell
NBER Working Paper No. 32768
August 2024
JEL No. C0

## ABSTRACT

Deep learning provides powerful methods to impute structured information from large-scale, unstructured text and image datasets. For example, economists might wish to detect the presence of economic activity in satellite images, or to measure the topics or entities mentioned in social media, the congressional record, or firm filings. This review introduces deep neural networks, covering methods such as classifiers, regression models, generative AI, and embedding models. Applications include classification, document digitization, record linkage, and methods for data exploration in massive scale text and image corpora. When suitable methods are used, deep learning models can be cheap to tune and can scale affordably to problems involving millions or billions of data points.. The review is accompanied by a companion website, EconDL, with user-friendly demo notebooks, software resources, and a knowledge base that provides technical details and additional applications.

Melissa Dell
Department of Economics
Harvard University
Littauer M-24
Cambridge, MA 02138
and NBER
melissadell@fas.harvard.edu

# Deep Learning for Economists

MELISSA DELL[*]

*Deep learning provides powerful methods to impute structured information from large-scale, unstructured text and image datasets. For example, economists might wish to detect the presence of economic activity in satellite images, or to measure the topics or entities mentioned in social media, the congressional record, or firm filings. This review introduces deep neural networks, covering methods such as classifiers, regression models, generative AI, and embedding models. Applications include classification, document digitization, record linkage, and methods for data exploration in massive scale text and image corpora. When suitable methods are used, deep learning models can be cheap to tune and can scale affordably to problems involving millions or billions of data points.. The review is accompanied by a companion website, EconDL, with user-friendly demo notebooks, software resources, and a knowledge base that provides technical details and additional applications.*

Deep neural networks have led to many recent scientific achievements – ranging from landing a rover on rugged Martian terrain to creating capable chatbots to transforming the diagnosis of disease. Deep neural networks typically map *unstructured* data - such as text, document image scans, satellite and other imagery, videos, and audio - to a continuous vector space. In the above examples, these vectors are used to compute the instructions to steer the spacecraft, autoregressively predict what word comes next given a prompt, or identify whether an image contains a tumor. Analogously, an economist might use neural networks to detect the presence of informal vendors in street view images, or to measure the topics or people mentioned in firm filings or government documents.

At its core, deep learning is an approach for learning representations of data from empirical examples (LeCun, Bengio and Hinton, 2015). These representations simplify high dimensional unstructured data into continuous vectors. Deep neural networks learn representations at multiple layers of abstraction, combining non-linear neural network modules that each transform the representation in the previous layer of the neural network into a slightly more abstract representation, using learned weights (the term “deep” signifies these many layers of transformation). These weights are estimated by minimizing a loss function that compares model predictions for some task to ground truth examples.

Why would one use a neural network to transform the raw data into these vector representations, versus just working directly with raw texts or images? First, deep neural networks don’t just learn from the problem at hand. Rather, they incorporate relevant information in their parameters from exposure to massive-scale data. During pre-training, a modern language model, or vision model, will have been exposed to many millions of texts or images, learning the basic structures of language or vision. Exposure to vast amounts of data is essential to strong performance when processing unstructured data, because human language and vision are remarkably complex.

[*] Department of Economics, Harvard University and NBER, melissadell@fas.harvard.edu. I would like to thank four anonymous referees, the editor, and Jake Carlson for their helpful comments and suggestions. Yiyang Chen provided excellent research assistance.

This principle is called *transfer learning* and is core to the success of deep neural methods.

Moreover, raw pixels or words lack *context*, which is necessary for interpreting their meaning. Deep neural networks provide a powerful method for computing *contextualized representations*. They map terms or pixels to vectors that depend on other nearby terms or pixels, with parameters learned mostly through massive-scale pre-training.

Finally, raw texts and images are computationally unwieldy. In contrast, there are extremely optimized tools for continuous vector computations. For example, Silcock et al. (2023) make $10^{14}$ exact vector similarity calculations on a single mid-range GPU in 3 hours. This means that data can be analyzed at an *unprecedented scale*. Theories are tested with data, and while more data won't solve challenges of causality, in general, it will provide economists with more fine-grained information for testing various hypotheses.

This review aims to bridge the gap between state-of-the-art deep learning research and economic applications. It focuses on imputing low-dimensional structured data from unstructured texts or images, in contexts where the ground truth is uncontroversial but extraction needs to be automated due to the massive scale of the problem. This structured data is then used for causal or descriptive analyses, whether as an outcome, endogenous variable, instrument, or control. Tasks that economists already perform by hand or with traditional methods (*e.g.,* record linkage, text classification, document scan digitization) can be more accurately automated at scale, and deep learning also facilitates the extraction of novel data. Like a prior review by Gentzkow, Kelly and Taddy (2019), this review emphasizes text as data, but with methods developed since the publication of that article.

Many of the applications in this review fall under the broad umbrella of *classification*: mapping high-dimensional unstructured data into discrete classes. The classes could identify types of objects present in a satellite image or numbers and words in a document image scan. Alternatively, classes could identify the topics of texts, the underlying source they were reproduced from, or unique individuals or locations referenced in them. A language model can be used to encode the raw text into lower-dimensional dense vector representations, one for the full text and one for each individual term (where "dense" means that the vector has a non-zero value in every position). The researcher can use these vector representations to predict whether the text is about a given topic, which locations are referenced, etc., by adding a classifier layer to the language model. Image classification works analogously. Generative AI models can also be prompted to impute these classifications. Alternatively, one can work directly with the dense vector representations, which are referred to as *embeddings*.

Figure 1 provides a flow chart for approaching classification. The first question to ask is: "Are the classes enumerated ex ante?" Sometimes, classes are not known, or the researcher may wish to add classes when applying the model to new settings without re-training it. A classifier computes a score for each class using the last layer of the language or vision model. Therefore, it can only be estimated when the classes are specified and seen in training. If classes are not specified, the researcher will need to work directly with the embeddings. If there are many classes, *e.g.,* as with record linkage, where each unique entity can be thought of as a class, the researcher will again need to work with embeddings due to computational constraints in estimating a classifier.

When classes are specified ex ante and modest in number, either a classifier or generative AI may be well suited to the problem. If applications diverge from the data used to pre-train the neural network—common when working with historical data, document scans, or certain specialized settings—there is significant *domain shift* from the pre-training corpus, and it may be necessary to tune a customized clas-

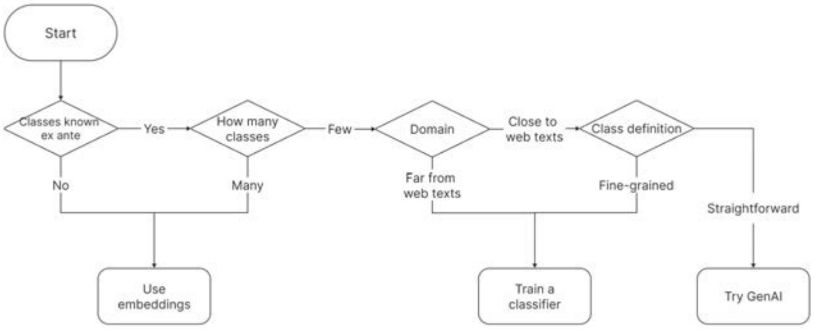


FIGURE 1. FLOWCHART FOR APPROACHING CLASSIFICATION.

sifier to achieve strong performance. The nuance of the class definitions is also important. For straightforward tasks, framing classification as text generation using an off-the-shelf generative AI model like OpenAI's GPT may work well. For more nuanced tasks, a custom-trained classifier can better capture that nuance by being exposed to fine-grained examples. If in doubt, a researcher can try an off-the-shelf method and switch to a customized classifier if performance is unsatisfactory. This review shows that while custom-trained classifiers most often outperform GPT on text classification tasks, generative AI and custom classifiers both perform well on straightforward tasks. The review also considers the costs of these approaches.

Table 1 summarizes this review's applications. Most can be framed as classification problems, and the article also reviews regression, where a neural network is used to impute continuous values from text or images.

TABLE 1—APPLICATIONS

| **Problem** | **Modality** | **Application(s)** | Section |
|---|---|---|---|
| ***Classifiers and GenAI*** | | | |
| Sequence classification | Text | Classify news article topics | VI.3 |
| Token classification | Text | Tag people, locations, orgs | VI.4 |
| Paired text classification | Text | Text b entails a? | VI.5 |
| ***Embedding Models*** | | | |
| Link structured data | Text, Images | Link firms, products, locations | VII.2 |
| Link unstructured data | Text | Link people mentions to Wikipedia | VII.3 |
| Classification w/ unknown categories | Text<br>Images | Track content dissemination; data exploration | VII.4 |
| Retrieval | Images | Optical character recognition | VII.5 |
| ***Regression*** | | | |
| Object detection | Images | Detect document layouts | VIII |

*Note:* Applications covered in this review.

In the pre-deep learning era, problems in different domains were approached in very different ways, using rules heavily engineered to the specific features of a given language or a particular type of image, etc.; whereas deep learning has a remarkable capacity to generalize. Natural language processing (NLP), computer vision, and audio processing all use the same state-of-the-art neural network architecture, for instance. This generalizability is apparent in the diverse applications discussed in this review.

A variety of neural network applications are beyond the scope of this review. It does not cover how language models can be used more generally to enhance economists' productivity, as discussed by Korinek (2023). It also does not cover machine learning methods outside of deep learning, such as those applied to structured data (which typically use shallower networks), as summarized in a review by Athey and Imbens (2019). Additionally, it does not examine using deep neural networks to compute approximate solutions to combinatorial optimization and high-dimensional DSGE problems. Approximating these solutions requires learning a neural network to map the original problem to a continuous vector space that preserves the essential properties of the problem. This is useful because computing an approximate solution in this space is dramatically faster than with traditional methods, allowing much larger problems to be approximated. There are many parallels with the methods covered in this review, but the applications are different enough that they necessitate their own treatment. Readers may wish to consult courses by Fernández-Villaverde (2024) and Vitercik (2023). Finally, a small literature directly uses deep neural networks in a causal framework. For instance, Lynn, Kummerfeld and Mihalcea (2020) use classifiers and experiments to examine how variations in texts causally influence decision-making. While there is a role for this when experimentally manipulating text, often a researcher would like to extract low-dimensional representations from high-dimensional unstructured data (*e.g.,* a text's topic, objects in a satellite image, numbers in a table scan, which textual records refer to the same firm) and use these—not the unstructured data—in the causal estimating equation. Hence, the focus here is on predicting these low-dimensional characteristics. The review does not attempt to summarize how these predictions have been used in economic studies, as this literature is new and rapidly evolving.

The reader may be wondering how quickly this review will become outdated. It is helpful to consider the popular metaphor that neural networks are like Legos: different neural network components can be configured in various ways to achieve different ends, or to achieve a more state-of-the-art version of the same end. This review focuses on frameworks where it is straightforward to swap in new neural network components as the literature advances, *e.g.,* replacing an older convolutional neural network with a vision transformer (Dosovitskiy et al., 2020), or updating a BERT language model backbone (Devlin et al., 2019) with the most recent language model. Technical and implementation details—those most likely to change as the literature advances—are provided on the accompanying EconDL website: `https://econdl.github.io/`. It provides a knowledge base organized into core topics, as well as links to open-source packages geared towards economists and pipelines that construct large-scale datasets with deep learning. Interested readers will find lecture notes and links to blog posts, textbook treatments, open courseware, and original papers. EconDL also links to demo notebooks for many of the applications in this review. The website will be updated on a continual basis, and some of the packages explicitly support swapping in new neural networks as the literature advances.

This article is organized as follows: Section I provides an overview of deep learning, and Section II introduces foundational architectures. Section III discusses data requirements of deep learning, Section IV considers bias and uncertainty quantification, and Section V addresses reusability and reproducibility. Next we turn to applications. Section VI introduces classification problems where the classes are defined ex ante and there are not too many classes, comparing classifiers and generative AI. Next, Section VII delves into embedding models, which are useful when the number of classes is large or the classes are not specified ex ante. Section VIII considers regression problems. There are other ways to approach the applications covered in this review. Section IX highlights why the methods emphasized are most likely to be suitable to the constraints faced by academic researchers. Section X concludes.

## I. An Overview of Deep Learning

### 1. What is deep learning?

Deep neural networks learn representations of raw data that extract information useful for specific tasks. Deep learning uses neural networks with many layers to map raw data to these representations, simplifying high-dimensional unstructured data into continuous vectors.

To represent data meaningfully for a given task, nodes (the numbers in a vector representation) in one layer of the neural network are transformed into nodes in the next by combining them with a non-linear function whose weights are learned parameters. These parameters—numbering millions to billions—are estimated by minimizing a cost function that compares model predictions on some task (*e.g.,* predicting masked terms in text) to ground truth examples. For those unfamiliar with neural networks, I recommend the introductory videos by Sanderson (2017).

The development of novel architectures and methods has made it feasible to optimize neural networks with millions to billions of parameters. These advances, while largely beyond the scope of this relatively broad review, are discussed in the EconDL knowledge base. In particular, many pioneering contributions in estimating deep networks were made in the literature on convolutional neural networks and are discussed in that post in the knowledge base.

Training a deep neural network from scratch requires a massive amount of data, and a couple of large-scale datasets are mainstays of the literature: ImageNet—a 14 million image dataset for image classification and related tasks (Deng et al., 2009)—and crawl corpora (*e.g.,* Cleaned Colossal Common Crawl (Raffel et al., 2019; Dodge et al., 2021))—massive public domain text datasets that essentially take a snapshot of the internet. Commercial models behind an API may also license proprietary training data. Training a deep neural model from scratch can require up to millions of dollars in compute, but fortunately, this is rarely necessary.

Deep learning has transformed many fields because of the power of *transfer learning*: deep networks trained in one domain can be adapted to many other domains with far fewer empirical training examples (often a few hundred to a few thousand) than would be required to train a model from scratch. For example, a researcher who needed to train a topic classifier could go to Hugging Face—the central hub for NLP (Section III)—and download a pre-trained state-of-the-art language model made publicly available by entities such as Google, Meta, and Microsoft. The language model was trained on a colossal corpus to predict tokens (words or sub-words) randomly masked from text. From this training, it learned to produce meaningful contextualized vector representations of texts. The researcher could add a classifier layer to the pre-trained language model and fine-tune the resulting neural network on their classification task using a relatively modest amount of labeled data. Most of the millions of model parameters would remain unchanged, as the model's basic understanding of language doesn't need updating, but the parameters that are of greatest relevance to the task at hand will update to improve model predictions (Merchant et al., 2020).

One of the striking findings that has emerged from deep learning in recent years is that returns from increasing model size can continue to accrue even with very large models (e.g., billions of parameters) (Raffel et al., 2020). Human vision and language are complex, and a rich expressive model is needed to capture that complexity. For example, suppose we wish to perform the simple task of classifying which of ten objects

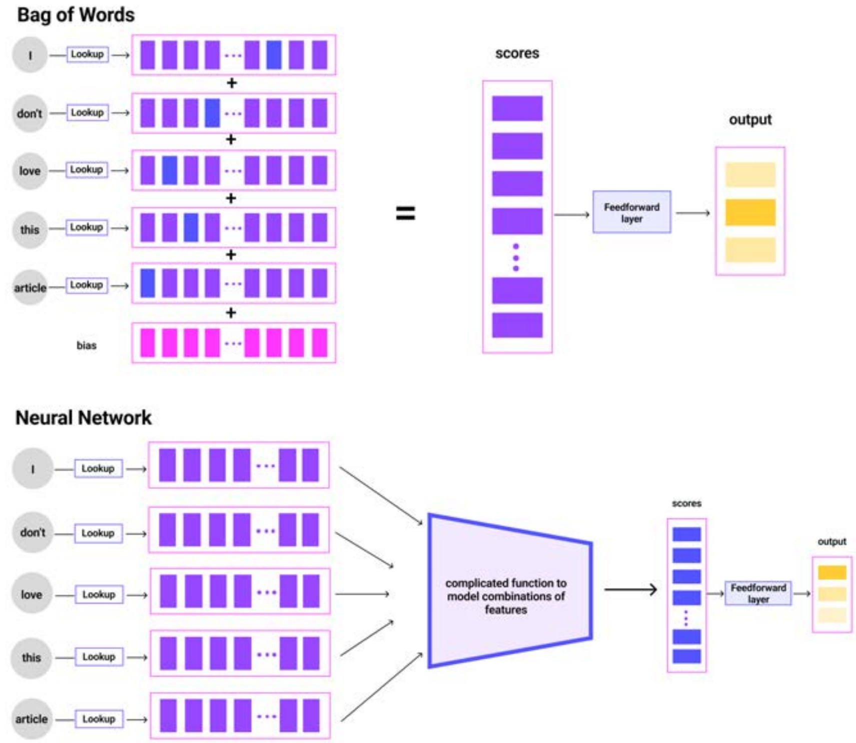


FIGURE 2. CLASSIFYING TEXT.

(a horse, car, etc) is contained in an image. The inputs to a classifier are the RGB values of each pixel $x_{1,1}, x_{2,1} \ldots x_{n,n}$. Suppose we estimate a linear classifier using these inputs. The score for each class $j = 1 \ldots 10$ is $\beta_{j,1,1}x_{1,1} + \beta_{j,1,2}x_{1,2} \cdots + \beta_{j,n,n}x_{n,n} + \gamma_j$, where the $\beta_j$ and $\gamma_j$ are estimated parameters. The classifier predicts assignment to the class with the highest score. The weight parameters for a class will be larger for pixels where the object in that class tends to be located. This is inherently brittle, because for each class there is only one parameter per pixel, but the horse could vary in its pose, its size, its position in the image, its color or build, etc. In the plot of the $\beta_{\textbf{horse}}$, one may see a horse standing in the middle of the image with two heads facing in either direction, as the linear classifier struggles to assign high values to pixels where horses are empirically likely to be located. A large neural network effectively allows for many such "filters" in predicting whether a horse is in the image.

Alternatively, suppose we would like to analyze whether statements (*e.g.,* in survey data) have a positive, negative, or neutral sentiment. A traditional approach, commonly used in the economics literature, is the bag-of-words method. The researcher looks up the sentiment of each word in a lookup table and aggregates these together to measure the sentiment of the sentence (Figure 2).

It is straightforward to see the limitations of this approach. Consider the following sentences: 'I love this article,' 'I don't love this article,' 'I don't hate this article,' 'There's nothing that I don't love about this article.' One cannot capture these different sentiments—even in these very simple sentences—by adding up independent representations of each word. Instead, we need to model nonlinear combinations of words, and neural networks are the state-of-the-art tool for approximating complicated nonlinear functions.

When processing text with a modern language model (bottom panel of Figure 2), a tokenizer first maps each word in the input to a number assigned by a lookup dictionary (if the word isn't in the dictionary, it gets split into subwords that are). These numbers get converted into vectors using learned parameters and are then passed through the neural network, which transforms them incrementally at each layer into semantically rich representations of the input tokens. Vision models are broadly analogous, taking pixels or patches of images as inputs.

The main alternative to neural networks is to use human-engineered features. In other words, the researcher pre-specifies rules for processing raw information. For example, table digitization could be automated by writing rules to detect the connected white space that separates rows and columns. With deep learning, the model is instead shown annotated examples of table layouts. The deep learning revolution has illustrated over and over—across many different tasks—that learning from empirical examples greatly outperforms human-engineered feature extraction in processing unstructured data. Some of this evidence is discussed in the EconDL knowledge base.

Deep learning is likely to outperform feature engineering in many economic applications as well. The information that economists would like to process is frequently complex and noisy. For example, noise is introduced into document scans through aging, scanning, and historical printing techniques; alternatively, text data may contain OCR errors or typos. Human language is complex, with many different ways to express the same sentiment and words that can change meaning significantly depending on the context. Noise and complexity create exceptions to human-engineered rules, which must also be hard-coded, and likewise there are exceptions to the exceptions. What initially seems like a simple task can quickly become convoluted as the researcher tries to hard-code these exceptions. Even if the results are satisfactory, the human-engineered system is likely to be heavily tailored to the case at hand and will not translate well to other data with different types of complexity and noise.

Another potential advantage of deep learning is that recipes for training and implementing neural networks are standard and reproducible, whereas significant discretion is inherent in human-engineered feature extraction. Even leaving aside researcher degrees of freedom, significant domain knowledge is required to engineer rules. For example, in statistical machine translation, large numbers of researchers worked over decades to engineer complicated statistical rules for machine translation. These systems were outperformed by a neural network that a few researchers developed over a few months. Subsequent advances in neural translation led to the Transformer architecture (Vaswani et al., 2017), which has since revolutionized natural language processing, computer vision, audio, and other domains.

## *II. Foundational Deep Learning Architectures*

This section provides a brief introduction to neural network architectures. For readers who are not familiar, I recommend consulting the EconDL knowledge base and the resources available there for a more detailed treatment.

### *1. The Basics of Neural Networks*

A neural network consists of layers of interconnected nodes, which are called *neurons*. Each neuron holds a value. The value of a neuron is computed by combining values from neurons in the previous layer using an activation function and learned weights. These many layers transform the input (*e.g.,* tokenized text) into vectors that are useful for performing a desired task.

An example activation function is rectified linear unit (ReLU): $f(x) = max(0,x)$, where

$$(1) \qquad x = w_1 \cdot i_1 + w_2 \cdot i_2 + \ldots + w_n \cdot i_n + b$$

$w_1, w_2, \ldots, w_n$ and $b$ are the learned weights and bias terms and $i_1, i_2, \ldots, i_n$ are the input values to the neuron from the previous layer in the network. When we feed data into a neural network, the input values are transformed by the activation functions at each layer. Nodes in the final layer are the output.

Activation functions are an important component of neural networks, because they introduce non-linearity, enabling the network to capture non-linear relationships in the data. The Convolutional Neural Networks post in the knowledge base provides further introduction to activation functions.

To optimize the neural network, the output is compared to ground truth labels using a loss function. What these labels measure depends on the objective: *e.g.,* predicting masked terms for a language model or predicting image classes for an image model. As with any optimization problem, we need to know the gradient of the loss with respect to each weight to minimize the function. For each layer starting from the output layer and moving backward through the network to the input layer, we compute the gradient of the loss with respect to the weights in that layer. This requires using the chain rule. The chain rule allows us to compute the derivative of the loss with respect to any weight in the network by multiplying together derivatives computed layer by layer. This is known as *backpropagation.* Weights are adjusted using a gradient descent algorithm.

Readers who are not familiar with backpropagation are encouraged to consult Sanderson (2017) for a high-production-value, graphical introduction. Readers wishing to gain a deeper understanding may enjoy Karpathy (2022), an advanced backpropagation tutorial. Nielsen (2015) provides a textbook treatment for those with no prior familiarity with neural networks. Goodfellow, Bengio and Courville (2016) offers a textbook treatment for those already familiar who would like an in-depth review, and Stevens, Antiga and Viehmann (2020) is aimed at those who would like to learn key concepts through hands-on implementations in PyTorch.

In a vanilla feedforward neural network, all neurons in one layer connect to all neurons in the next layer. Deep fully connected networks are rarely used in practice. Rather a few types of neural networks have dominated the deep learning literature. This review focuses on convolutional neural networks (CNNs; Section II.2), recurrent neural networks (RNNs; Section II.3), and the transformer (Sections II.4, II.5, and II.6).

### 2. *Convolutional Neural Networks*

CNNs leverage the spatial structure present in images and played a central role in ushering in the deep learning revolution. Despite the advent of a newer architecture for image processing—the vision transformer (Section II.6)—they remain widely used, can obtain near-state-of-the-art performance when appropriately modernized, and can be lighter weight and easier to tune than vision transformers. This section provides a brief introduction. I recommend that those unfamiliar with CNNs consult the short graphical introduction to convolution by Sanderson (2020), as a visual introduction to the concepts described below is particularly helpful. Additional resources are on the ‘Convolutional Neural Network’ page of the EconDL knowledge base.

Vision problems start with an image of a given height (in pixels), width (in pixels), and depth (*e.g.,* 3 for an RGB image). Convolutional layers are the core building block of a CNN. The layer’s parameters consist of a set of learnable filters, *e.g.*, 3 × 3, 5 × 5, or 7 × 7 weight matrices. These filters are only applied to the nodes immediately surrounding a given node when computing the output for the next layer and extend through the full depth of the input. Each filter is convolved (moved) across the input, producing an activation at each spatial location. Using the same weights for different spatial locations drastically reduces the number of parameters compared to fully connected layers. Moreover, parameter sharing ensures that features can be detected regardless of their position in the image. This gives CNNs a degree of *translation invariance*, desirable since *e.g.,* a horse is a horse regardless of its position in the image.

The locality bias inherent in small convolutional filters is logical because the interpretation of a pixel is more influenced by its neighboring pixels than by distant ones. Despite the localized nature of these filters, a CNN still achieves an extensive receptive field through the depth of the network. CNNs are adept at learning hierarchical features: lower layers, which have a more limited receptive field, capture simple patterns such as edges, while deeper layers capture increasingly complex structures.

In addition to convolutional layers, CNNs also use pooling layers. If $M$ different convolutional filters are applied to a neural network layer, the depth of the next layer will be $M$, since each filter produces an activation for each spatial location. Pooling layers reduce this depth, preventing the number of parameters from becoming infeasibly large. Often, a CNN consists of alternating convolutional and pooling layers.

The central challenge to estimating neural networks with many layers is the vanishing gradient problem (Bengio, Simard and Frasconi, 1994). Backpropagation computes the gradient of the cost function with respect to each weight in the network. This requires applying the chain rule to find the gradient of the loss with respect to the output of each layer, and then the gradient of the output of each layer with respect to its input. Derivatives can become very small for extreme values of their inputs. Backpropagation multiplies small gradients together. Hence, the gradient may become exponentially smaller as it flows back to the earlier layers. If the gradient becomes extremely small for the initial layers, learning will be very slow or stop altogether. The post on 'Convolutional Neural Networks' on EconDL examines the evolution of CNN architectures, including key innovations that allowed for the optimization of much deeper, more expressive networks, circumventing the vanishing gradient problem (Krizhevsky, Sutskever and Hinton, 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016; Xie et al., 2017; Howard et al., 2019; Liu et al., 2022).

## 3. *Recurrent Neural Networks*

CNNs require fixed-size inputs, as neural networks are initialized with weight matrices of fixed dimensions (variably-sized images need to be resized or padded). In contrast, *recurrent neural networks* (RNNs) are designed to process variably-sized inputs and outputs. They historically played an important role in NLP (Hochreiter and Schmidhuber, 1997; Greff et al., 2016), though they have since been superseded by the transformer. While researchers generally should use a transformer for NLP applications, I introduce RNNs as a point of comparison to the transformer.

RNNs process a sequence of inputs—*e.g.,* tokens in a text—iteratively. At each time step, they maintain a state that captures historical information about the input sequence. This state is updated iteratively as the network processes each element in the sequence, allowing the network to 'remember' previous elements in the variable-length input.

Long-range dependencies are important to human language. To use a prominent example from Vaswani et al. (2017), 'The animal didn't cross the road because it was too tired' versus 'The animal didn't cross the road because it was too wide.' Does 'it' refer to the animal or the road? This depends on dependencies between 'it' and other tokens in the input. The most prominent RNN is the bi-directional LSTM (Long Short-Term Memory) (Hochreiter and Schmidhuber, 1997). Bi-directionality captures dependencies in both directions by feeding the input sequence forwards and backwards. Readers can find a more detailed introduction to LSTMs in the EconDL knowledge base.

## 4. *The Transformer*

The *transformer* (Vaswani et al., 2017) has revolutionized NLP and made inroads in nearly all areas of deep learning, including vision, audio, graphs, and reinforcement learning. For readers who are not familiar, I recommend the Illustrated Transformer blog post by (Alammar, 2018*b*), widely recognized as the most accessible introduction. An annotation of the original paper by Rush (2018) is also a classic reference.

The original transformer was a neural translation model whose key ingredient is *attention*. All tokens (words or sub-words) in a sequence are fed into the model in parallel, and the model attends to all other tokens in the context to create contextualized representations for each token. Contextualized representations contrast with traditional static representations of words (Mikolov et al., 2013; Pennington, Socher and Manning, 2014; Olah, 2014), where a given word in a training corpus always has the same representation. The transformer solves the locality bias of RNNs—where information is lost as the hidden state is passed along to each sequential token—because it can attend to any token in the sequence, whether nearby or not.

Self-attention is quadratic, which limits the length of text that can be passed into the model at one time (the context window). A typical context window length in open-source models is 512 tokens. For many problems, this is sufficient (*e.g.,* the texts can be chunked, or the first 512 tokens are sufficient to form a meaningful document representation). There are also models with sparse attention mechanisms that allow for long context windows.

Inputs are fed into the transformer in parallel, rather than sequentially as in an RNN, allowing training to fully leverage the parallel computing power of GPUs. This makes it computationally feasible to train much larger models on more data for longer, all of which improve performance (Raffel et al., 2020).

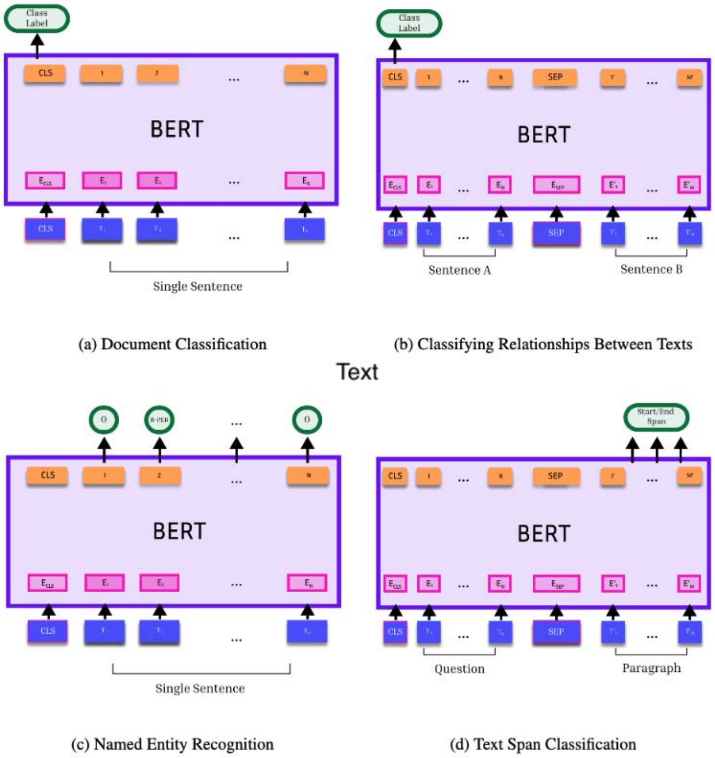
## 5. *Transformer Large Language Models*

Most modern NLP applications use transformer-based large language models (LLMs). For those unfamiliar with LLMs or needing a refresher, I highly recommend Jay Alammar's 'Illustrated GPT-2/3' (Alammar, 2019, 2020) and 'Illustrated BERT' (Alammar, 2018*a*) blog posts, which provide an intuitive graphical introduction to transformer LLM architectures.

There are two main types of transformer LLMs. *Generative (decoder) models* predict the next word in a sequence (Radford et al., 2019; Brown et al., 2020). They are typically used for text generation. Since they are trained by predicting the next word, they can only attend to prior tokens when creating contextualized representations for a given token. This is called *causal attention*.

In contrast, *masked (encoder) language models* are bidirectional: in creating contextualized representations of words in a sequence, they can attend to all words in the sequence (*masked attention*). The model is trained by predicting masked tokens (Devlin et al., 2019; Liu et al., 2019; Sanh et al., 2019*a*; Lan et al., 2019; He et al., 2020). Encoder models are typically used when a researcher aims to create representations of text that entail feeding an entire text to a model. Bidirectionality is helpful for such tasks, because the context both before and after a word is useful for creating semantically meaningful representations of it. Language models can also combine encoder and decoder transformer blocks, *e.g.,* Raffel et al. (2019).

With the transformer architecture, the same pre-trained language model can be used as the "backbone" for a wide variety of tasks, facilitating transfer learning. This is illustrated in Figure 3, which is adapted from an illustration in the original BERT paper (Devlin et al.,



(a) Document Classification

(b) Classifying Relationships Between Texts

Text

(c) Named Entity Recognition

(d) Text Span Classification

FIGURE 3. TASKS PERFORMED WITH A TRANSFORMER ENCODER LANGUAGE MODEL.

2019). A transformer language model produces a vector representation for each token (word or sub-word) in its input, as well as a <cls> representation that represents the entire input text. The first panel shows that full text sequences can be classified by adding a classifier head to the <cls> token. The classifier is a feedforward neural network that aggregates the nodes in the <cls> vector of the final transformer layer into a score for each class, using learned weights. Alternatively, two texts can be embedded jointly - separated by the special token <sep> - and then a classifier can be added to the <cls> token to classify the relationship between texts (panel b). Or, each individual token can be classified (*e.g.,*tagging whether it refers to an individuals, location, etc.) by adding classifier heads to each of the token embeddings (panel c). Spans of text can also be identified (*e.g.,* the answer to a question; panel d).

A variety of different transformer pretrained language models are detailed in the EconDL knowledge base post on ‘Transformer Language Models.’

## 6. *Vision and Audio Transformers*

The transformer has transformed many other areas of deep learning, including computer vision (Dosovitskiy et al., 2020; Touvron et al., 2021; Grill et al., 2020; Caron et al., 2021; Ali et al., 2021; He et al., 2022; Chen, Xie and He, 2021). Vision Transformers (ViTs) use the same transformer architecture as transformer language models, with some adaptations to make them suitable for images. Unlike in NLP, where transformer language models have greatly outpaced the prior technology, in vision the gains from the transformer relative to CNNs are more modest (Liu et al., 2022). An appropriately modernized CNN will often be competitive with a similarly sized ViT. Moreover, the smallest CNNs (*e.g.,* Howard et al. (2019)) are smaller than lightweight ViTs (*e.g.,* Mehta and Rastegari (2021)) at present and can also perform well on straightforward tasks. Practically, I recommend starting with a lightweight CNN—which will be easier and significantly cheaper to train and deploy—and examining the performance of a larger CNN or ViT model if the lightweight model is inadequate. The EconDL post on Vision Transformers provides more details about ViT architectures.

The extent to which the transformer can be used to process highly diverse types of unstructured data is strikingly illustrated by its application to audio. State-of-the-art performance was achieved by applying a ViT to the spectrogram image of the audio (Gong, Chung and Glass, 2021). Even more strikingly, performance was maximized by pre-training on ImageNet—the main benchmark for vision that consists of over 14 million natural images (*e.g.,* of dogs, food, etc.)—a powerful illustration of transfer learning even across modalities.

### 7. *Optimizing Neural Networks*

Being able to optimize a neural network is clearly central to using them in research. The optimizer (see *e.g.,* Kingma and Ba (2014); Goh (2017)), initialization (He et al., 2015), and normalization (Ioffe and Szegedy, 2015; Santurkar et al., 2018) are all important. To estimate deep neural models, the researcher must also select various hyperparameters (*e.g.,* Li et al. (2017); Falkner, Klein and Hutter (2018)). Interested readers are referred to the post on ‘Basics of Training and Optimizing Neural Networks’ in the EconDL companion knowledge base. The packages on EconDL choose reasonable defaults for various hyperparameters in an effort to make training neural networks more user-friendly.

While there are various details at play, my main practical takeaway from guiding many students in optimizing neural networks is that when performance is unexpectedly poor, it is most often due to either a poorly chosen learning rate or incorrectly formatted input data. The learning rate determines the size of the steps taken by the optimizer while adjusting weights during training. If it is too low, the model won’t update; if it is too high, the model weights will oscillate wildly. Moreover, data are expected in a specific format, and transformations may be performed on the fly (for instance, many neural networks require fixed-size inputs). Neural networks learn from empirical examples, and unexpectedly poor performance is often the result of feeding them misformatted examples.

I also recommend that those new to deep learning train and deploy models using a cloud server specifically designed for this purpose. The EconDL tutorials use Google Colab. Installing deep learning packages locally requires resolving extensive dependencies, which may be challenging for those with limited experience. For more experienced users who heavily utilize deep learning in their research, purchasing their own GPUs can often provide significant cost savings.

## III. *Training Data*

High-quality training and evaluation data are integral to the utility of deep learning. In *supervised learning*, data are partitioned into *labeled* and *unlabeled* sets. The unlabeled set, which consists of all the data that the deep learning model will be applied to, is typically much larger than the labeled set. Deep learning is *self-supervised* when relevant labels are gleaned automatically from the data itself. For example, language models can be pre-trained by predicting words that have been randomly masked from a massive text corpus. Analogously, a masking strategy can be applied to images for the self-supervised pre-training of vision models (He et al., 2022). Self-supervised learning is most commonly used in pre-training, and then the pre-trained model is transferred to another domain and applied to unlabeled data (potentially following additional supervised tuning on a modest amount of data from the target domain). Finally, in *unsupervised learning*, there are no ground truth labels, as the goal is to discover underlying structures in the data, grouping them according to similarities. Embeddings can be clustered, for example, to discover these relationships.

Supervised methods are common in economic applications, as the goal is often to use neural networks to extract some characteristics from unlabeled data. Economists might also continue self-supervised pre-training. This is most common when the domain of their application shifts considerably from the domain that the model was pre-trained on Gururangan et al. (2020). For instance, an economic historian analyzing 18th century legal texts might first continue pre-training the language model on that database (by predicting masked tokens), in order to impart better understanding of 18th century legal jargon.

Unsupervised learning is most useful for data exploration. In empirical economics, the norm is to specify a narrowly defined hypothesis and test it statistically, which often lends itself to supervised applications. However, methodical data exploration using unsupervised methods can be a powerful tool for gleaning stylized facts from novel unstructured data.

When conducting supervised learning, labeled data are further divided into *training data*—used to train the model, *validation data*—used to tune model hyperparameters or select prompts, and *test data*—used only to compute the model evaluations that will be reported in the findings. The researcher should always have a high-quality, representative test set to evaluate model performance. If the data used to evaluate model performance are not representative of the unlabeled dataset—and in particular, if some appreciable portion of the unlabeled data has no support in the labeled data—model performance on the evaluation data may diverge widely from model performance on the unlabeled data, which is the underlying object of interest.

In an ideal scenario, representative test sets can be created through random sampling. However, this is not always possible, particularly when classes that the researcher would like to measure are highly imbalanced. Suppose that a researcher needs to extract texts on a topic of interest from a massive web corpus, and the relevant topic appears only once in every ten thousand texts. The labeling requirements for sampling enough positives randomly are clearly infeasible. This scenario is common in social science, where researchers frequently need to classify relevant information from a massive corpus—*e.g.,* media or government documents—where only a tiny share of content is about the topic of interest.

While strategies to draw the most informative samples to annotate have generated a large machine learning literature on *active learning* (EconDL provides a detailed discussion in the context of text classification), there is little work on selecting representative samples for training, evaluation, or debiasing when class imbalance is severe. Discriminative active learning (Gissin and Shalev-Shwartz, 2019) selects samples to label that maximize the difficulty of distinguishing between the labeled and unlabeled data and can work well with relatively balanced data. It does not work well with severe class imbalance because it fails to sufficiently sample the rare class(es). Other active learning approaches seek to sample near the decision boundary of a classifier, which will sample the rare class(es) and can maximize predictive accuracy. However, this will provide an unrepresentative sample.

Social scientists, instead, frequently use the presence of certain keyword(s) to choose content to label. However, by construction, this fails to place positive sampling probability on all instances, increasing the odds that some types of unlabeled data have no support in the labeled data. This can generate prediction bias that is systematically correlated with the error term in the downstream causal estimating equation—where the researcher intends to use the deep learning model predictions—since semantics and omitted variables often both vary across space and time.

Embedding models (Section VII)—deep neural networks that create a space where distances between vector representations of texts or images are meaningful—provide a metric that can be used for stratified sampling of data to label for training or evaluation. The closer a text/image is to a set of queries about a class (*e.g.,* 'this article is about tax policy'), the higher the probability that it comes from that class, making distances in this space useful for stratified sampling. A stratified sampling approach can also provide informative negatives for training: samples that a pre-trained model place near a query but that are not related to that query in the way the researcher intends. This is an active area of research, where economists have the potential to make important contributions. The EconDL site will update on this literature as it advances.

Training data need not be drawn from the same distribution as the unlabeled data—given the power of transfer learning—although predictive accuracy will typically decline with the magnitude of the *domain shift* between the target and training data. Oftentimes, datasets that already exist or can be extracted from web texts allow for the cheap creation of a much larger training set than the researcher could label by hand. High performance on a target dataset is then ensured by further tuning on a much smaller set of hand-crafted labels from the target data.

*Congruence labeling*—when two (or more) annotators label the same data points—is important for ensuring the quality of training and evaluation data. Even seemingly simple tasks are often messier than expected once taken to real-world unstructured data. Congruence labeling also ensures that annotators have understood the task and are producing high-quality labels. In challenging labeling tasks, researchers may wish for all labeled data to be double-annotated, resolving discrepancies by hand. In more straightforward cases, congruence labeling may only be necessary for a smaller subset of the data, to ensure that the task is well-defined and annotators have properly understood the instructions. In machine learning papers, the researcher is typically expected to report the congruence between annotators, as well as to publish annotator instructions, and this can be useful in economic applications as well.

### *IV. Bias and Uncertainty Quantification*

There are many limitations to using deep learning to solve social or economic problems (see, for instance, papers from the ACM Conference on Fairness, Accountability, and Transparency `https://facctconference.org/` and Cui and Athey (2022)). Here, our focus is much narrower: to impute or explore features of unstructured data that humans are likely to agree on, in contexts where the size of the raw dataset is orders of magnitude too large to extract features manually.

Caution is necessary when applying deep learning to contexts that require subjective judgment. For example, researchers have shown that the self-identified political orientations of annotators influences their incongruence on political sentiment labeling (Shen and Rose, 2021). Sentiment classification in the deep learning literature has been designed largely around laptop, restaurant, and movie reviews, contexts where there is typically an explicit sentiment about the product that can be validated by stars. In many applications that economists care about—such as sentiment in media data, political speeches, corporate reports, etc.—sentiment can be much more implicit, and humans may not agree on it. If a model is fed annotations that reflect the subjective biases of the annotator—versus a well-defined ground truth—or if it simply does not have enough examples because the distinctions to be made are complex, it will make inaccurate predictions that may be systematically biased. Models can also inherit biases from pre-training, and there are large literatures on bias and fairness in AI (Mehrabi et al., 2021). These challenges can be mitigated by sticking to straightforward tasks with a clearly defined ground truth.

Economists can make valuable contributions on uncertainty quantification, which is uncommon in much of the deep learning literature. Conformal inference can provide uncertainty quantification for prediction tasks. Facilitated by the collection of a ground truth calibration dataset, conformal methods produce prediction sets with marginal coverage guarantees under mild conditions. A canonical tutorial is Shafer and Vovk (2008); see Chernozhukov, Wüthrich and Zhu (2021), Cattaneo et al. (2022), and Lei and Candès (2020) for recent contributions.

Asymptotically motivated inference usually requires that estimates of model parameters be unbiased, which poses a problem for 'black box' machine learning predictors that typically trade off bias and variance to produce predictions with low mean squared error. A long literature in semi-parametric inference (*e.g.,* Robins, Rotnitzky and Zhao (1994)) has worked to remedy these issues, culminating in a large, recent econometrics literature on debiased machine learning (*e.g.,* Chernozhukov et al. (2018); Chernozhukov, Newey and Singh (2022)).

There are many parallels between the debiased machine learning literature in econometrics and a literature on 'prediction powered inference' in the deep learning space (*e.g.,* Angelopoulos et al. (2023); Zrnic and Candès (2023)). Broadly speaking, imputing structured characteristics from unstructured data (the focus of the deep learning literature) and causal inference (the focus of the econometrics literature) are special cases of a more general problem of imputing missing data. In causal inference, potential outcomes are missing, whereas in many deep learning prediction applications, low-dimensional structured characteristics are missing because it is prohibitively costly to extract them from high-dimensional unstructured data manually. The prediction powered inference literature examines how deep learning predictions can be debiased using a high-quality auxiliary sample of ground truth labels for the population of interest. This information is used to measure the bias induced by imputation, which is then corrected, ultimately allowing the researcher to perform valid inference without sacrificing the information available from using a model pre-trained on a larger dataset that makes biased predictions. The deep learning model is treated as a black box. One can show that the prediction powered inference framework is equivalent to debiased machine learning.

## *V. Reusability and Reproducibility in Deep Learning*

Deep learning has been built to a remarkable degree upon open science and open data, although recent years have seen a pronounced shift towards proprietary models and data as the commercial potential of the technology has become increasingly clear. Nevertheless, the amount of open resources is staggering, and deep learning would not have made the strides it has without widespread sharing of models and datasets. Given the centrality of transfer learning and massive-scale pre-training, the field as we know it would not exist without open science.

The more economics can create an open science culture around big data, whenever data privacy concerns allow, the more we can benefit as a profession from the positive externalities of transfer learning. For example, deep learning researchers are often incentivized to share their code on GitHub as soon as possible, as a way of staking claim to their contribution, or to release a dataset as soon as it is constructed so that a fast-moving literature will use it for longer. Moreover, publication venues in deep learning often require compliance with agreed-upon metadata and ethical frameworks for data and code release (Gebru et al., 2021; MLCommons, 2024; Mitchell et al., 2019; Holland et al., 2018). While

I would not advocate that economics wholesale adopt these standards, it is worth considering whether there are standards for model and dataset release that could facilitate reproducibility and reusability of deep neural models in economics.

The largest hub for deep learning models and data is Hugging Face. A wealth of language models and text data can be found there, some examples of which are examined in the demo notebooks linked through EconDL. Hugging Face recently acquired timm, a central repository for vision models, making Hugging Face a one-stop shop for many language and vision tasks.

## *VI. Classifiers*

Having provided an introduction to deep learning, I now turn to applications. Classification is frequently integral to economic analyses. In the era of big data, a researcher may first need to extract relevant data using classification. For instance, they might start with a massive-scale corpus of news, social media posts, earnings calls, or legislative records and need to extract only coverage about interest rates, immigration, or higher education out of millions or even billions of texts in the full corpus. This much more limited corpus is then used to extract the measure(s) to be used in some downstream causal estimating equation. While this step often receives scant attention, biased classification will result in selection bias into the sample used in the downstream causal estimating equation, potentially significantly biasing the conclusions. Alternatively, a researcher might impute structured data—*e.g.,* geographic locations mentioned in texts, their sentiments or topics, or what type of objects appear in a satellite image—using classification.

This section first introduces classifiers (Section VI.1), as well as describing the use of generative AI for classification (Section VI.2). Then, it introduces *sequence classification*, in which a class label is imputed for a sequence of text: *e.g.,* a sentence, paragraph, or document (Section VI.3). It compares the performance of custom-trained classifiers to generative AI on 19 different text classification tasks. Classification can also be applied to individual terms in a text (Section VI.4). Finally, a classifier can be used to compare texts to each other (Section VI.5). I focus on text classification for ease of exposition. Image classification–of a full image or pixels or objects within an image–is analogous, using a CNN or vision transformer rather than a language transformer. It is covered in depth in the EconDL knowledge base post titled 'Convolutional Neural Networks.'

### *1. An Introduction to classifiers*

In traditional classification, a neural network predicts a score for each of $N$ classes, and the input is assigned the class with the highest score. For those unfamiliar with classifiers, Sanderson (2017) provides an excellent graphical treatment of classification in the context of classifying images of digits.

Recall the analogy that neural networks are like Legos. Central to the power of transformer models is the ability to use the same pre-trained language model as the backbone for a wide variety of classification tasks. This is illustrated in Figure 3. A transformer language model produces a vector representation for each token (word or sub-word) in its input, as well as the `<cls>` representation that summarizes the entire input text. The text sequence can be classified by adding a classifier head to the `<cls>` representation (panel a). The classifier is a feedforward neural network that aggregates the nodes in the `<cls>` vector into a score for each class using learned weights. As shown in Figure 3, panel c, individual tokens can likewise be classified by adding classifier heads to their vector representations. Alternatively, two texts can be jointly embedded and then a classifier can be added to the `<cls>` representation to classify the relationship between them (panel b).

Training a classifier is one of the most straightforward tasks in deep learning. The open-source package `LinkTransformer` can be used to train text sequence classifiers, with a demo available via EconDL. While the base transformer language model could be frozen when training a classifier, and various layers of the transformer could be used as inputs to the classifier, typically all parameters are allowed to update, with the classifier layer attached to the final layer of the transformer.

Classifier training is a supervised task, and the model must see a sufficient number of examples from each class during training in order to perform well on unlabeled data. When creating labels for classification, the labeled data should be relatively balanced across classes (*e.g.,* positive and negative samples in the case of binary classification).

To train a classifier, we also need an appropriate loss function. The two most common losses for classification are Support Vector Machine (SVM) loss, also referred to as hinge loss, and cross-entropy loss.

Given a sample with true label $y_i$ and the score vector $\mathbf{p}$ for the class scores produced by the neural network, the SVM loss is:

$$L_i = \sum_{j \neq y_i} \max(0, p_j - p_{y_i} + 1) \tag{2}$$

The loss sums over the incorrect classes, imposing a penalty if the score of the correct class is not at least some threshold amount above the score(s) for the incorrect class(es). The threshold can be set to one without loss of generality, as it just scales the learned weights.

Cross-entropy loss measures the dissimilarity between the predicted score distribution and the true distribution. Consider a neural network used for a multi-class classification problem with $C$ classes. Let $y$ be the true label of a sample, represented as a one-hot encoded vector. For a sample belonging to class $i$, $y_i = 1$ and $y_j = 0$ for $j \neq i$. Let $\mathbf{z}$ be the raw numbers (often referred to as logits) produced by the neural network for that sample. A classification layer will produce one score for each class. The predicted score of class $i$, obtained using the softmax function, is:

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{3}$$

While the class scores are frequently referred to in the literature as ‘probabilities’, they are not probabilities in a statistical sense. How peaky they are will depend on the regularization of the neural network.

The cross-entropy loss between the true label and the predicted distribution is:

$$\mathrm{CE}(y, p) = -\sum_{i=1}^{C} y_i \log(p_i) \tag{4}$$

As $y$ are one hot vectors, this simplifies to:

$$\mathrm{CE}(y, p) = -\log(p_{\text{true class}}) \tag{5}$$

where $p_{\text{true class}}$ is the predicted probability of the correct class.

With SVM loss, once the score for the correct class surpasses a threshold, it is indifferent to elevating the score of the correct class further. On the other hand, the cross-entropy loss pushes the correct class towards 1. This means that during the early stages of training, accuracy might increase suddenly without a significant change in the loss. In real-world scenarios, both losses typically yield comparable results.

Binary classifiers are evaluated using the F1 score, a metric that combines recall (true positives divided by true positives plus false negatives) and precision (true positives divided by true positives plus false positives).

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Perfect precision and recall yield an F1 score of 1, whereas the worst score is 0. F1 is a harmonic mean of precision and recall and thus tends to be closer to the smaller of these two metrics. If either the precision or the recall is low, the F1 score will also be low. F1 is preferred over accuracy because if classes are imbalanced, accuracy can be high simply by always predicting the majority class.

### 2. *Generative AI for classification*

Large generative AI models like GPT, Claude, or Llama (commonly referred to in the literature as *foundation models*) use a decoder transformer architecture (Section II.5) to autoregressively generate text given a prompt. In practice, they might also be connected to an external database (*e.g.,* the internet) via a retrieval-augmented language modeling (RALM) setup (the post on retrieval in EconDL contains more information about RALM). In a fundamental sense, these models are performing classification, autoregressively predicting the most likely next token in a discrete vocabulary at each time step. By default, models like GPT are stochastic; they predict the next token from a distribution of the most probable tokens.[1]

To perform classification tasks using generative AI, the user needs to prompt the model. Prompting a generative language model is, in many ways, less straightforward than tuning a classifier via gradient descent, as the space of discrete prompts is infinite and prompting has generated a large and unwieldy literature.

A few clear insights are though worth emphasizing. Centrally, prompt tuning should be done on a validation set, never on the test set used to evaluate performance. The latter may overfit the prompt to the idiosyncrasies of the test set, making performance on it unrepresentative of performance on the unlabeled data.

A literature on chain-of-thought prompting suggests breaking tasks down into simple steps, making them more digestible (Wei et al., 2022). This aligns closely with my experience, where simple prompts work much better than lengthier and more detailed ones. If a problem requires lengthy instructions, try to break it down into multiple problems, prompting the model at each step. There is also literature on demonstrating tasks for generative LLMs (*e.g.,* Khattab et al. (2022)). Whether this is useful will depend on the nature of your task, and I recommend checking whether demonstration helps using a validation set. Liu et al. (2023) provide a review of prompt engineering. For autoregressive models like GPT, they recommend *prefix* prompts; *e.g.,* "I love this class. What's the sentiment of this review?" This contrasts with cloze prompts: "I love this class, it is a $[z]$ class."

This paper examines the performance of GPT-3.5 and GPT-4o on topic classification of historical newspaper articles as of June 2024. Over the past year, I have performed this exercise with older models, as well as GPT-4 and GPT-4 Turbo. GPT-4 and GPT-4o perform similarly, edging out Turbo. I haven't seen any systematic improvements with new releases. Your mileage may vary, as there undoubtedly are tasks where the larger, newer (and hence more expensive) models will perform better.

I also examined two other leading AI models, from Anthropic: Claude Haiku and Claude Opus. These led to significantly worse performance (with F1 scores typically 10-40 points lower than GPT) and are not reported due to space constraints. The drivers of this lower performance are twofold. First, Claude refused to produce an output for texts that it assessed as harmful. A distinguishing feature of Claude is its 'Constitutional AI' framework, which sets out certain ethical principles (*e.g.,* harmlessness necessitates that responses should be peaceful, ethical, and avoid content that might be considered offensive in non-Western cultures). Some articles on past conflicts (mostly objectively reporting on events) and a range of other topics (*e.g.,* content about the introduction of contraception written in the 1960s) were considered harmful. Moreover, Claude didn't always generate the de-

[1] At the time of writing, setting `top_p` to 0 makes GPT deterministic.

sired Yes/No format, making it impossible to extract whether the article was on-topic. Neither of these behaviors arose with GPT. Perhaps this could be fixed with more prompt tuning, or it may change with a future update.

Regardless, with a classifier it is often fairly straightforward to interpret why it makes particular errors and how to fix them (by adding more training data for the types of instances it is confusing), whereas GenAI at present can feel more like a black box. There is more going on under the hood, with models trained with reinforcement learning to produce responses deemed desirable by the commercial entities training the models. This may or may not pose a problem for a given academic application and underscores the importance of rigorous evaluation with a test set.

The pros of generative AI for classification are: 1) startup costs are low, requiring minimal programming expertise or understanding of what is going on under the hood, and 2) it can be used *zero-shot* (without the user providing training data), whereas tuning a classifier requires training data. We will see that, if anything, custom-trained classifiers tend to have a performance edge on text classification, though the most straightforward tasks can be performed very well with generative AI. This is consistent with the broad consensus that human-level performance can generally be attained on supervised tasks given adequate, high-quality training data.

This brings us to potential disadvantages of generative AI. Using a large model behind an API does not provide the same fine-grained control as training a classifier. While models such as GPT allow users to expose the model to empirical examples, for the experiments in this paper, this did not lead to improvements in performance. It is not fully understood how demonstration through prompting conditions these models, whereas it is clear how providing training examples updates a classifier through gradient descent. Not all models learn equally when exposed to the same amount of training data (as demonstrated empirically in Section IX), and lightweight models like those used with customized classifiers tend to update very efficiently. A custom classifier also has an edge when it comes to interpretability and reproducibility. Results from a commercial API may no longer be reproducible if a model is deprecated, and as discussed above, commercial GenAI models can be more of a black box. These concerns could be mitigated by using an open-source foundation model, such as Meta AI's Llama (Touvron et al., 2023). The startup costs and hardware requirements, however, negate the ease-of-use advantage. Finally, classifiers using a lightweight backbone such as RoBERTa (Liu et al., 2019) are cheap to deploy over a massive number of texts, whereas commercial models at present can be very expensive. This may change if competition increases and research on cheap deployment advances.

To decide whether a classifier or generative AI is most suitable for a task, I would recommend first doing a back-of-the-envelope calculation to ensure that generative AI is within budget. If so, create test and validation sets, tune prompts, and evaluate its performance. If performance is not adequate, a training set to tune a custom classifier will be necessary. If the user knows ex ante that guaranteeing reproducibility is imperative, that there is substantial domain shift from web texts, or that the task requires fine-grained control, they might go straight to training a custom classifier. Data privacy requirements can add an additional layer to consider for those working with confidential data.

### 3. *Sequence classification*

Economists might wish to impute a variety of structured information at the level of a text: *e.g.,* its topic, the type of content it contains, or its sentiment. To illustrate text sequence classification, this section trains 19 different binary topic classifiers, applied to massive-scale databases of historical news (Dell et al., 2023; Silcock et al., 2024), and compares them to generative AI. It would have been very difficult for annotators to keep 19 different topic definitions in mind to create multi-class labels; hence, binary classification is used. Binary classifiers cannot be combined into a multi-class classifier, as negatives for one topic may be positives for another. Keeping prompts simple for generative AI also suggests binary classes.

The annotated data were congruence-labeled by highly skilled annotators, with discrepancies resolved by hand. Congruence labeling ensured high-quality data and facilitated the development of a well-specified definition. For example, in the case of the crime classifier, annotators disagreed on whether articles about Watergate should be considered on-topic, and a zero-shot model showed a massive spike in crime coverage in 1974 due to Watergate. While potentially a reasonable definition, I did not want crime coverage to be skewed by political scandals (which receive massive coverage), and the classifier quickly learned this with a modest number of labels.

A frequent question is how many labels are required. This will vary. Topics that are more diverse or require learning a more complex definition will require more labels. Topics that were seen many times in the pre-training of the language model may require fewer labels. Fortunately, training a classifier is compute-efficient. If, after the first round of training, results are unsatisfactory, it is straightforward to add more labels and retrain. An error analysis may give a sense of what types of texts require more training examples. Since labeling is costly, we recommend starting with fewer labels and adding more if needed.

Table 2 provides the split statistics for the various topic classification tasks examined in this section.[2] Labeled data are randomly split into training, validation, and test data. Validation data are used to select hyperparameters, select the model checkpoint (when to stop training), and to tune prompts, whereas the test data were used only to compute Table 2. The prompts for Table 2 are listed in the supplemental materials.

The classifiers were trained with `LinkTransformer`, which supports using any base language model available on Hugging Face. We used DistilRoBERTa (82M parameters) (Sanh et al., 2019*b*) and RoBERTa large (335M parameters). RoBERTa (Liu et al., 2019) is a widely used, improved version of BERT. Distilled language models are smaller models that are trained to match the performance of a larger model. The distilled version runs faster but often with a performance loss. We used a consistent set of hyperparameters across classification tasks, which appear to work well more generally (a learning rate of $1e-6$ or $1e-5$ and a batch size of eight).

In most cases—across a diversity of topics—the tuned classifier tends to outperform or equal the performance of GPT, though for the more straightforward tasks, GPT's performance can be very good, particularly in the case of GPT-4o (GPT-4 performed similarly). The training data used to produce these classifiers are high quality. With lower-quality labels, such as those created with online annotation platforms where quality is notoriously poor, a custom-trained classifier may well be consistently worse than GPT. These comparisons may also change in the future.

More generally, generative AI performs best on straightforward topics that it was likely extensively exposed to during pre-training. The further the domain shifts from training data—primarily modern web texts—the more performance deteriorates. For horoscopes, obituaries, and articles on the polio vaccine—all extremely straightforward—GPT performs nearly perfectly (as do the classifiers). However, there are also topics for which GPT performs poorly; for example, politics, a topic that is challenging because it is broad and diverse, with content drawn from the late 19th and early 20th centuries and including both local and national politics. World War I has an F1 score from both GPT models in

[2] The politics classifier is taken from a published paper (Dell et al., 2023) with different aims, and hence the split shares and overall number of annotations differ somewhat.

Table 2—F1 on test set and summary of train-test split

| **Topic** | **F1 on test set** | | | | | **# of labels** | | |
|---|---|---|---|---|---|---|---|---|
| | GPT-3.5 | GPT-4 | GPT-3.5 Trained Model† | Distil RoBERTa | RoBERTa Large | Train | Eval | Test |
| advice | 0.72 | 0.85 | 0.55 | 0.87 | **0.97** | 319 | 68 | 68 |
| antitrust | 0.85 | **0.94** | 0.84 | 0.92 | **0.94** | 329 | 70 | 70 |
| bible | 0.52 | 0.81 | 0.10 | 0.85 | **0.87** | 314 | 67 | 67 |
| civil rights | 0.59 | **0.87** | 0.54 | 0.85 | **0.87** | 943 | 202 | 202 |
| contraception | 0.83 | 0.91 | 0.72 | 0.88 | **0.97** | 597 | 127 | 127 |
| crime | 0.85 | 0.80 | 0.85 | 0.85 | **0.90** | 463 | 98 | 98 |
| horoscope | **1.00** | **1.00** | 0.92 | 0.96 | **1.00** | 288 | 61 | 61 |
| labor movement | 0.77 | 0.90 | 0.79 | 0.89 | **0.94** | 253 | 54 | 54 |
| obituaries | 0.98 | **1.00** | **1.00** | 0.96 | **1.00** | 272 | 57 | 57 |
| pesticide | 0.58 | 0.91 | 0.71 | 0.89 | **0.98** | 873 | 187 | 187 |
| polio vaccine | 0.92 | **0.99** | 0.94 | 0.96 | 0.97 | 350 | 74 | 74 |
| politics | 0.67* | 0.62* | 0.74 | **0.86** | 0.85 | 2,418 | 498 | 1,473 |
| protests | 0.74 | 0.81 | 0.79 | 0.90 | **0.91** | 351 | 75 | 75 |
| Red Scare | 0.81 | 0.86 | 0.79 | 0.90 | **0.91** | 1,852 | 396 | 396 |
| schedules | 0.79 | 0.95 | 0.81 | 0.95 | **0.96** | 346 | 74 | 74 |
| sports | 0.80 | 0.92 | 0.88 | **0.94** | **0.94** | 339 | 72 | 72 |
| Vietnam War | 0.91 | 0.94 | 0.98 | 0.98 | **0.99** | 738 | 157 | 157 |
| weather | 0.94 | 0.92 | 0.94 | 0.94 | **0.95** | 569 | 57 | 57 |
| World War I | 0.72 | 0.74 | 0.51 | 0.89 | **0.92** | 690 | 164 | 192 |

*Note:* †This column reports the F1 for trained models (based on either DistilRoBERTa or RoBERTa-Large, whichever works better) using labels generated by GPT-3.5.

* The results with asterisks were produced on a random sample of 500 out of total 1,473 articles in the test set.

the low 70s, much worse than the Vietnam War, which likely has greater representation in the training corpus. Moreover, language has changed more since World War I, translating into greater domain shift. Yet with minimal labels, the RoBERTa classifiers can adjust to this domain shift.

At present, GPT is likely to be well beyond the budget of most social science researchers for large corpora, though I do not cite figures here as prices fluctuate and could change considerably depending on competition and technological advancement. In contrast, training a RoBERTa classifier on the number of labels shown here is very cheap (at the time of writing, it could be done within minutes on a $9.99/month Google Colab plan or a mid-range Nvidia GPU card). I have also had students, with patience, train similar models on a laptop, though getting access to a decent GPU through cloud compute or dedicated hardware is preferable. Deploying the classifiers, even across millions of articles, is also cheap, and can be done either with cloud CPUs or on a mid-range GPU card in hours. One can circumvent the expense of generative AI by training a classifier on labels predicted by GPT. Table 2 shows this can work when GPT produces very high-quality labels. However, where GPT performs less well, training on noisy data can magnify errors.

Figure 4, drawn from Silcock et al. (2024), takes binary classifiers that apply across time and deploys them to a dataset of 2.7 million unique newswire articles published between 1878 and 1977. Various trends are evident, such as the Prohibition-related crime coverage spike in the 1920s or the surge of Civil Rights and protests coverage in the 1960s.

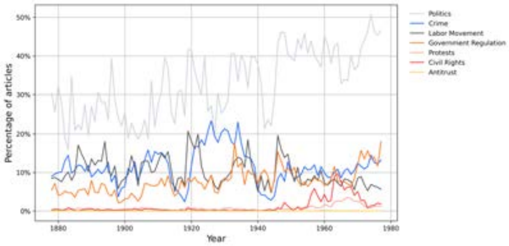


FIGURE 4. SHARE OF NEWSWIRE ARTICLES WITH A GIVEN BINARY TOPIC TAG.

It is worth saying a word about how neural methods compare to sparse methods for text classification, which (with varying degrees of sophistication) rely keywords. For example, one common sparse method is TF-IDF: Term Frequency (TF) is the raw count of term $t$ in document $d$. Inverse Document Frequency (IDF) measures the importance of a term in the entire corpus. If a word appears in many documents, it's not a good identifier of a given document. The IDF of a term $t$ for a corpus $D$ is given by $\log \frac{\text{Total number of docs in } D}{\text{Number of docs with term } t \text{ in } D}$. The TF-IDF score for a term is simply the product of its TF and IDF scores. The higher the TF-IDF score, the more important a term is to a specific document relative to the context of the entire corpus. To rank documents from a corpus based on their similarity to a query using TF-IDF, each document and query is represented as a sparse, high-dimensional vector, with each dimension corresponding to a unique term from the corpus. The weight of each term in the vector is its TF-IDF score. The angle between any two vectors captures the similarity between the texts. One can think of this as akin to a keyword search that downweights terms that appear across the corpus.

We refer to methods like TF-IDF as sparse because each term in the corpus forms a dimension in the vector space. Most terms in the vocabulary will not be present in a single document, leading most entries in the term vector to be zero.

Sparse methods are useful when exact term overlap is highly informative. However, relying on term overlap is often a major shortcoming, as language is complex. There are many ways to say the same thing, and the same term can have different meanings. Moreover, noise (*e.g.,* typos, OCR errors, abbreviations) is ubiquitous. Semantics vary across time and space, as do many omitted variables. This could result in correlation between prediction error and the error term in the causal estimating equation where the keyword predictions are used. Moreover, while terms can be mined, more often they are simply chosen, creating a researcher degrees of freedom problem.

Neural methods address these shortcomings by using a large language model to map texts to a dense vector representation, *e.g,* a 768-dimensional vector composed of non-zero terms. The dimensionality of this vector depends on the base language model. The pretrained language model is imbued with language understanding, and hence dense methods account for contextual and semantic similarities. This allows them to generalize over synonyms and semantically similar phrases, and to be more robust to other noise.

Dell et al. (2023) compare the neural classifier for politics, shown above, to mined keywords as well as keywords suggested by ChatGPT. Neural methods lead to significantly more accurate predictions.

## 4. *Token classification*

Researchers may need to extract information about individual terms in a text, rather than the text as a whole. This problem is analogous to sequence classification, except that classifier heads are added to the representations for each token in the final layer of the transformer, rather than only to the `<cls>` representation (Figure 3, panel c).

This section develops an example of token classification, named entity recognition (NER), which detects named entities in texts. These entities can be defined however the researcher desires, as long as a clear, consistent definition and sufficient labels for training exist. For example, the researcher may want to identify locations referred to in social media posts. Alternatively, a researcher extracting family relationships from obituary data might wish to tag the relationships of individuals to the deceased (child, parent, sibling, officiant, etc). Or, a researcher wishing to convert biographical texts into a structured dataset might tag birthplace, mother, father, university, spouse, and employer.

NER is a classic task that has generated a very large literature, and there are many open-source pre-trained models and datasets on Hugging Face. In this literature, entity classes typically include person, location, and organization. CoNLL is a prominent benchmark (Sang and De Meulder, 2003) used to pre-train various models on Hugging Face. WNUT is another (Nguyen et al., 2020), which focuses on noisy user-generated texts (tweets). A researcher will need to create labels if they depart from the entity types emphasized by the benchmarks. NER typically uses BIO labeling - the first token in an entity is labeled B (for begin), the following tokens are labeled I (for interior), and tokens that are not of interest are labeled O. If the entity types of interest are people (P) and locations (L), tags would be B-P, B-L, I-P, I-L, and O.

Figure 5 presents results from applying NER to historical newswire articles, plotting the shares of over 27 million entities that fall into four categories: person, location, organization, and miscellaneous. The results are sensible, for example showing a spike in location and miscellaneous named entities (*e.g.,* aircraft names) during World War II.

One can also ask generative AI to recognize entities in text and convert the output into a table. As with sequence classification, researchers can test whether performance is adequate for their needs by constructing representative validation and test sets.

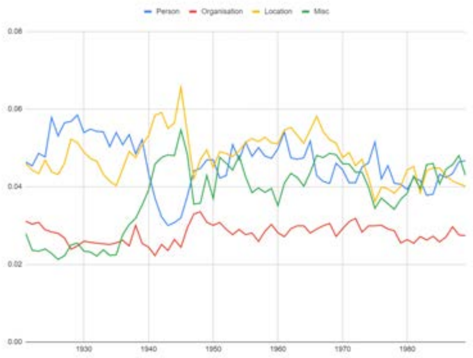


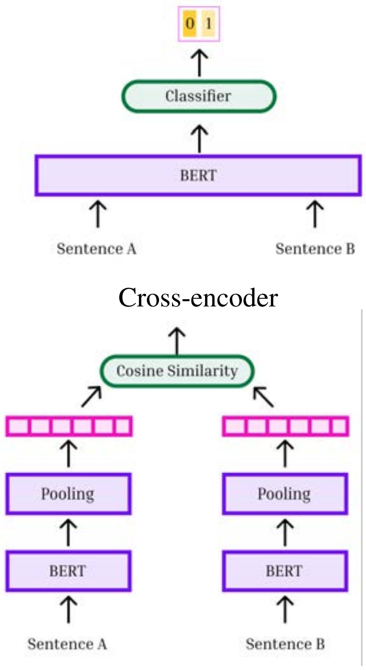
FIGURE 5. SHARES OF ENTITY TYPES IN NEWSWIRE ARTICLES.

### 5. *Relationships between texts*

There are a variety of contexts in which we would like to measure whether two texts are related in some pre-specified way. For example, we could phrase topic classification as a task in which we would like to classify whether one statement entails another: does the article text entail the statement "this article is about monetary policy"? Alternatively, are two texts noisy duplicates of each other? Do they take the same stance on a political issue? Does one follow the other?

Figure 6 illustrates two approaches to comparing texts. The top shows a *cross-encoder*: two texts are concatenated, with a `<sep>` token between them. These texts are jointly passed into the transformer, and a classifier head classifies how they are related. This approach allows for full cross-attention between all tokens in each of the texts. The bottom figure illustrates a *bi-encoder* approach: texts are embedded separately, and then we compute the similarity between them using some distance metric, such as cosine similarity. Bi-encoders are very useful, and we elaborate on them in Section VII.

A cross-encoder allows for full cross-attention between terms. Because they are



Cross-encoder

Bi-encoder

FIGURE 6. ARCHITECTURES TO COMPARE TEXTS.

jointly embedded, the terms in the texts being compared attend flexibly to each other when creating representations. In contrast, a bi-encoder computes a single representation for each text separately, and then these representations are compared. Hence, cross-encoders tend to have higher accuracy. However, they also have significant drawbacks. Most centrally, if we need to compare $M$ texts to $N$ other texts, this would require embedding $M \times N$ texts. This quadratic cost quickly becomes infeasible, since each text is passed through a neural network with hundreds of millions of parameters.[3] In contrast, for the bi-encoder to compare $M$ texts to $N$ other texts, only $M + N$ embeddings are required, making this approach highly scalable. To get the best of both worlds, the literature often uses a bi-encoder to get the $n$ most similar texts to a query text (for some small $n$), and then re-ranks these matches with a cross-encoder.

## VII. *Embedding models*

To estimate a classifier, the classes must be specified ex ante, since the number of classes determines the number of parameters in the neural network, and classes must be seen in training. Prompting generative AI for classification tasks also entails specifying what the classes of interest are. However, there are a variety of problems where the classes are not known ex ante or where the researcher would like to add new classes later without having to retrain the model. Moreover, if the number of classes is large, it can become computationally intractable to compute the softmax over all the classes for the loss function.

These common scenarios can be addressed by working directly with the embeddings from the final layer of the transformer or CNN, rather than estimating an additional neural network layer (the classifier) that maps embeddings to class scores. This avoids the pre-specification of classes. Moreover, vector similarity calculations are highly optimized, allowing for problems at the scale of millions or even billions of classes.

This section introduces a variety of applications that can be approached with embedding models. Record linkage (*e.g.,* of individuals, firms, locations, or products across datasets) is a common task that can be framed as a classification problem with many classes (*e.g.,* each individual, firm, etc.). It is particularly amenable to embedding methods (Section VII.2). These methods can handle settings—like multilingual linkage or linkage with multiple, noisy text descriptions—that are very difficult to tackle with traditional string matching methods. Similar methods can be used to link mentions of individuals, firms, etc. in unstructured texts (*e.g.,* social or print media, firm filings, biographies, government documents) to an external knowledge base such as Wikipedia (Section VII.3). Tracking the

[3] Moreover, if we ask a cross-encoder to classify if $a$ is the same as $b$, $b$ is the same as $c$, and $a$ is the same as $c$, intransitivities may result.

spread of text or images through media is another application of potential interest, where the classes are often unknown ex ante (Section VII.4). In some cases, the aim may be exploratory, to uncover the stylized facts in a massive novel text or image dataset, and embedding methods are well-suited to such descriptive analyses. Finally, optical character recognition (OCR) can be framed as an image classification task, where the researcher might wish to add characters or words subsequently without retraining the model, suggesting embedding methods (Section VII.5).

Working directly with embeddings requires distances between vector representations to be meaningful. The geometric properties of pre-trained transformer language models are not well-suited to this task. For example, representations of low-frequency words are pushed outwards on the hypersphere. The sparsity of low-frequency words violates convexity, and the distance between embeddings is correlated with lexical similarity. This leads to poor alignment between the embeddings of semantically similar texts and poor performance when individual term representations are pooled to create an average representation for text sequences (Ethayarajh, 2019; Reimers and Gurevych, 2019).

Mathematically, the problem is that the embedding space created by a pre-trained transformer model is not *isotropic*, meaning that the representations are not evenly distributed. When embeddings are isotropic, no particular direction is favored. This uniform distribution ensures that the distances between vectors accurately reflect their relationships, making the space more effective for tasks that depend on these distances. Contrastive learning is a widely used method that improves isotropy and is discussed before turning to embedding model applications.

### *1. Contrastive learning*

Contrastive learning aims to learn similar representations for semantically similar inputs and dissimilar representations for semantically different inputs, where the definition of similarity is given by empirical training examples. The contrastive loss function encourages the model to reduce the distance in embedding space between positive examples (*e.g.,* similar texts or images) and increase the distance between negative examples (*e.g.,* dissimilar texts or images). Contrastive training reduces anisotropy (Wang and Liu, 2021), significantly improves pooled representations of text sequences (Reimers and Gurevych, 2019), and improves alignment between representations of semantically similar texts.

Contrastive learning follows the bi-encoder setup shown in Figure 6. Bi-encoders form a single representation for each instance, by passing it through a transformer and pooling (averaging) term level embeddings.[4] Representations can be compared by computing their vector similarity. In practice cosine similarity is frequently used since representations are on a unit hypersphere.

There are different options for the loss function depending on the training data. Contrastive loss (Chopra, Hadsell and LeCun, 2005) uses positive and negative pairs, incentivizing positives to have the same representation and negatives to be above a threshold distance apart. A cosine loss uses a continuous measure of the difference between instances. With triplet loss (Hermans, Beyer and Leibe, 2017), training data consist of triplets: an anchor, one positive example for the anchor, and one negative example. Embeddings of positives are incentivized to be more similar to the anchor than negatives. With the InfoNCE loss (Oord, Li and Vinyals, 2018), multiple negative examples are compared to a single positive example. Supervised contrastive loss (Khosla et al., 2020) generalizes InfoNCE to allow for multiple positive and negative examples. More details, including mathematical formulations, are provided in the ‘Contrastive Learning’ post in the EconDL knowledge base.

[4] Representations are pooled rather than using the `<cls>` token because this has been shown to have better performance (Reimers and Gurevych, 2019).

The metric space created by a neural network should be interpreted based on the contrastive loss function used to train the network. For instance, with contrastive loss, instances in the same class are incentivized to have similar representations, whereas different instances are incentivized to be above a threshold distance apart. Hence, local distances are meaningful, but global distances are not, since being more dissimilar beyond the threshold does not affect the loss.

When contrastively training on paired data, the weights of the two encoders used to embed each of the instances can be the same (a symmetric encoder) or different (an asymmetric encoder, as in Karpukhin et al. (2020)). Symmetric encoders provide a more parsimonious model, and hence require less data and compute to train. In practice, they can perform well even when two distinct types of instances are being encoded (*e.g.,* search queries and documents that contain their answers).

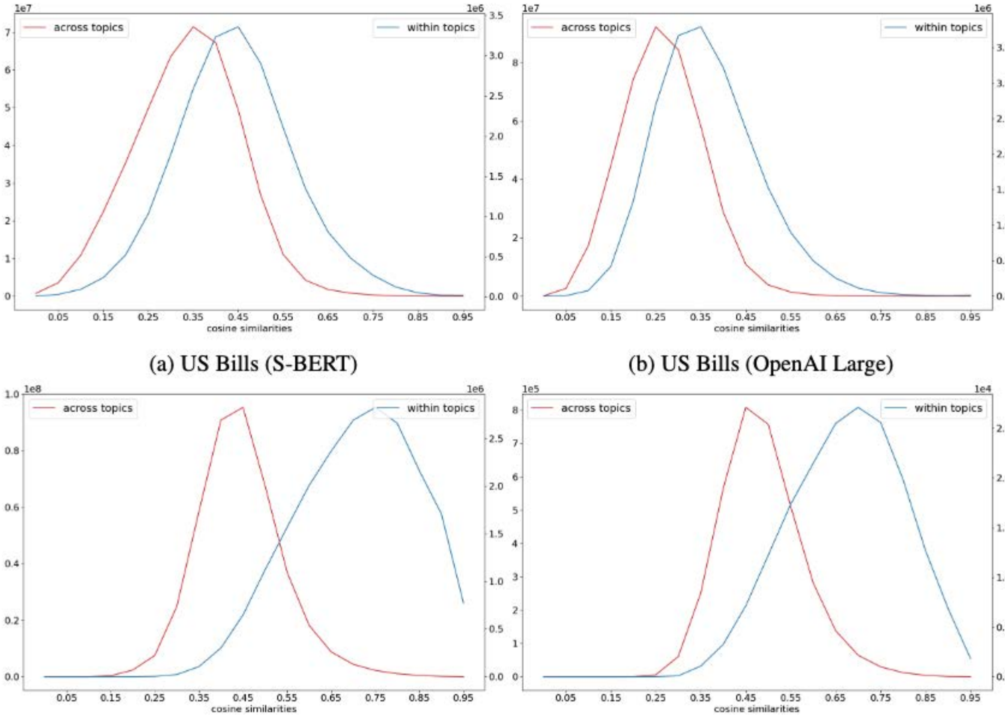
Selecting informative negative examples is important for contrastive learning. If the negatives are too 'easy'—*e.g.,* they are dissimilar in the embedding space of the pre-trained model used to initialize training—little is learned. In some contexts, the researcher can use prior knowledge to select 'hard' negatives for training. In other contexts, they can be mined by using a pre-trained model to choose negative examples with similar embeddings. This can work well if the researcher knows ex ante which instances are negative, as would be the case when training on synthetically generated data. When negatives are drawn from unlabeled data, however, mining hard negatives without a human in the loop risks inadvertently selecting positives. Sometimes training on random negatives may be sufficient, although this often requires large batch sizes (*e.g.,* GPU cards with a lot of memory), which is not amenable to academic compute budgets. This section's applications consider how negatives are selected.

Embeddings are also available off-the-shelf. Sentence-BERT (Reimers and Gurevych, 2019) is a prominent and well-supported open-source model. (In this literature, the term 'sentence' is used to refer to any text sequence, which could be a phrase, sentence, or entire document.) Moreover, OpenAI sells quite affordable sentence embeddings.

While there has been considerable interest in developing all-purpose embedding models that can excel zero-shot on any task (Cao, 2023)—and a larger model will on average outperform a smaller model zero-shot—fine-tuned lightweight embedding models have important advantages over zero-shot embeddings. Intuitively, embeddings provide a single representation for each text (or image). An off-the-shelf representation will capture lots of different information about the text/image. However, the researcher is typically interested in some narrowly defined aspects of it. Fine-tuning will accentuate the relevant dimensions, creating better separation between classes in embedding space.

Consider the following empirical example. I take the comparative agendas dataset on U.S. legislation (Wilkerson et al., 2023), which assigns topic tags to congressional bills, and calculate pairwise similarities between the embeddings of the legislative descriptions using three different models: off-the-shelf lightweight S-BERT embeddings (Figure 7, panel a), OpenAI large embeddings (panel b), and embeddings produced by tuning S-BERT on paired positives and (random) negative bills from a training split of these data (panel c). The blue distribution plots cosine similarities within topics, and the red line plots cosine similarities between topics. Identical representations have a similarity of 1.

With off-the-shelf models, embeddings are indeed more similar within than across topics (*e.g.,* the blue distribution is shifted to the right of the red distribution), but the differences are not stark. SBERT and OpenAI perform similarly. In contrast, once the model has been tuned on target data, embeddings are much more similar within topic than across topic, as the contrastive training accentuates the im-



(a) US Bills (S-BERT)

(b) US Bills (OpenAI Large)

(c) US Bills (Tuned)

(d) UK Acts of Parliament (Tuned)

FIGURE 7. EMBEDDING SIMILARITIES WITHIN AND ACROSS TOPICS.

portance of topic in determining how the language model maps texts to embedding space. Much of the overlap in distributions comes from edge cases where articles fall between topics or cover multiple themes. Panel (d) uses the model tuned on U.S. bills to compare embeddings of UK Acts of Parliament within and across topics. While there is some domain shift, there continues to be marked separation, showing the fine-tuned model's ability to generalize to similar problems.

### 2. *Record linkage with structured data*

Record linkage is central to many economic analyses. A researcher might need to link individuals, locations, firms, organizations, product descriptions, or academic papers across datasets. Traditionally, records have been linked using measures like Levenshtein edit distance—which counts the number of character insertions, deletions, and substitutions to convert one string into another—or Jaccard similarity—which computes the similarity between substring n-gram representations of strings. A recent machine learning literature focused on matching across e-commerce datasets shows the promise of transformer LLMs for improving record linkage. Yet at the time of writing, these methods have not yet made widespread inroads in social science, with rule-based methods continuing to overwhelmingly predominate (*e.g.,* see reviews by Binette and Steorts (2022); Abramitzky et al. (2021); Bailey et al. (2020)).

To make these methods more accessible, Arora and Dell (2024) designed

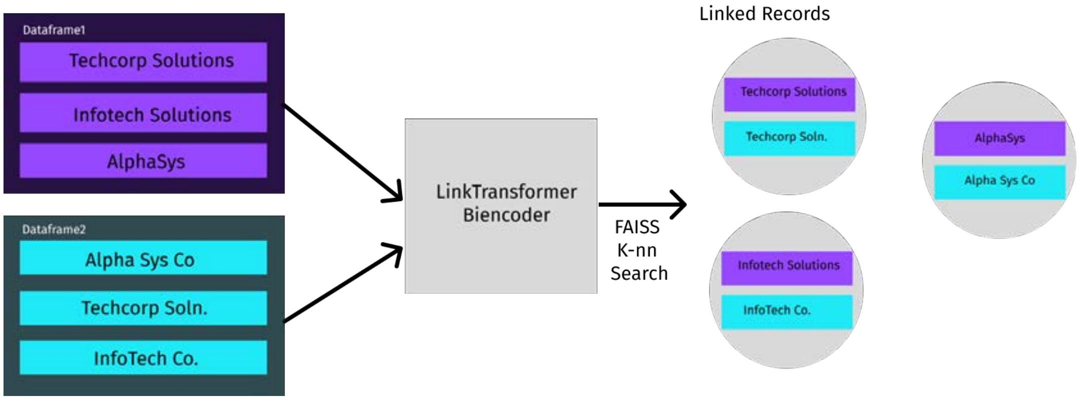


FIGURE 8. LINKTRANSFORMER ARCHITECTURE.

LinkTransformer, a package for using transformer models for record linkage that is geared towards social scientists. The study documents that transformers outperform traditional string matching methods across a variety of tasks and languages, often by a wide margin. Applications include linking 1940 Mexican tariff schedules and linking 1950 Japanese firm-level records using multiple noisy fields, as well as linking modern firms and products across six languages. A multilingual model can link products across languages without the need for translation.

This work was motivated by a variety of projects that I had to abandon in the pre-deep learning era because sparse methods performed poorly and hand-linking was infeasible. As with any predictive task, it is incumbent upon the researcher to evaluate whether performance is acceptable using a test set.

The LinkTransformer model architecture is shown in Figure 8. The texts that need to be matched are encoded using a transformer language model. For each query, LinkTransformer finds the nearest neighbors in the corpus, as measured by the cosine similarity between the embeddings. This is extremely fast, as it uses the highly optimized FAISS (Facebook Artificial Intelligence Similarity Search) backend (Johnson, Douze and Jégou, 2019). LinkTransformer returns a ranking as well as the cosine similarity scores, which can be used for 1-1, 1-many, or many-many merges, including no matches (captured when the similarity to the nearest record is below some threshold).

Just as traditional sparse methods like edit distance return distances between records, LinkTransformer computes a distance metric powered by all the semantic knowledge embodied in the pre-trained language model, as well as any additional knowledge gained through contrastive training. For example, ABC Corporation, ABC Co., and ABCC are very similar semantically—and hence nearby in embedding space—since 'Co.' and 'C' represent 'Corporation,' but these strings have a high Levenshtein edit distance. Examples like this are common in record linkage tasks, given the prevalence of abbreviations, different ways to describe the same product or firm name, OCR errors, and typos. Embedding similarity can be used analogously to how researchers use string distance metrics.

LinkTransformer seamlessly supports linking with multiple fields, serializing fields by concatenating them with a <sep> token that the package automatically selects to be compatible with the base language model tokenizer. The study provides an example of linking 1950s Japanese firms across different large-scale, noisy databases using the firm name, location, products, shareholders, and banks. This type of linkage problem would be highly convoluted with string matching methods, as fields are noisy (*e.g.,* products are described in different ways across datasets, different subsets of managers and shareholders are listed, etc.). A large language model can handle these challenges with ease because it captures semantic similarity.

`LinkTransformer` allows users to employ Sentence Transformer models, OpenAI embeddings, models tuned on the target task, or any transformer language model available on Hugging Face. The general picture that emerges, across 20 diverse linking tasks, is that models custom trained on a modest number of labels tend to perform best, followed by off-the-shelf embeddings from OpenAI, and then off-the-shelf Sentence Transformer models (though there is some variation from task to task). The results are consistent with the discussion of off-the-shelf versus customized embedding models above.

`LinkTransformer` also provides APIs to use transformer large language models for other data processing tasks, *e.g.*, classification, aggregation, and de-duplication, as outlined in tutorial notebooks linked on EconDL. Users can also find a 'Train Your Own `LinkTransformer` Model' tutorial, for when customization is necessary. Contrastive training requires both positive and negative pairs (in this case, linked records and distinct records). The user can provide only positives—in which case `LinkTransformer` chooses negatives at random—or can provide both positives and negatives, if hard negatives are available. To promote reusability, reproducibility, and extensibility, models can be shared to the Hugging Face hub with a single line of code.

When the task is to link scanned documents (as in much of economic history), computer vision may also be useful. In the vision only record linkage models developed in Arora et al. (2023), record linkage is OCR-free, using only the image crops of the firm names to be linked. This is explored in a challenging setting, linking firms between historical Japanese publications where one publication is written horizontally and the other vertically. In general, vision-only linkage leads to reasonably high accuracy. However, there are some matches that a vision model cannot resolve because a firm can write its name in different ways.

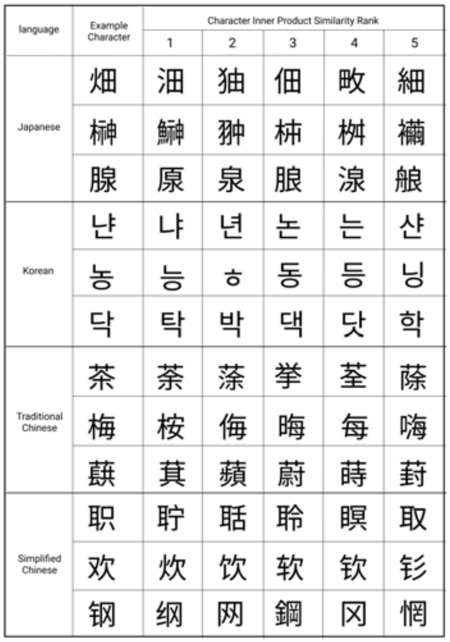
Embedding models can combine vision and language transformers, leveraging an understanding of both semantic and visual similarity. Arora et al. (2023) show that a multimodal model leads to extremely accurate linkage of OCR'ed Japanese firm-level records on customers and suppliers, whereas less accurate string distance metrics produce a different supply chain network that would likely lead to biased downstream economic analyses.

Combining text and image embeddings is most straightforward when the embedding spaces are aligned. In other words, image and text representations of the same thing need to have similar embeddings; *e.g.,* a picture of an avocado and the text 'avocado' have similar embeddings when an image and text encoder respectively map them to vector space. Arora et al. (2023) start with a Japanese version of CLIP, which stands for Contrastive Language-Image Pre-training. CLIP is an OpenAI model contrastively trained to align text and image encoders using 400 million image-caption pairs scraped from the web (Radford et al., 2021). Arora et al. (2023) do further pre-training on synthetically noised pairs of document crops and their corresponding OCR'ed texts. Pooled text-image representations are then used to link firms.

There are other ways to incorporate visual similarity into linkage that may be helpful when OCR errors are present. Approximate string matching methods count the number of edits (insertions, deletions, and substitutions) needed to transform one string into another (Levenshtein et al., 1966). In practice, not all string substitutions are equally probable, and efforts to construct lists that vary their costs date back at least to 1918, when Russell and Odell patented Soundex (Russell, 1918; Archives and Administration, 2023), a sound standardization toolkit that accounts for the fact that census enumerators often misspelled names according to their sound. Such methods can significantly improve the accuracy of edit distance linking in the contexts for which they are tailored but are labor-intensive to extend to new settings due to the use of hand-crafted features. This skews research with linked data—necessary to examine many economic questions—towards higher resource settings that are not representative of the diversity of human societies.

Yang et al. (2023) develop an extensible, self-supervised method for determining the relative costs of character substitutions in databases created with OCR. OCR often confuses characters with their homoglyphs, which have a similar visual appearance (*e.g.,* ’0’ and ’O’). Yang et al. (2023) augment digital fonts to contrastively learn a metric space where different augmentations of a character (*e.g.,* the same character rendered with different fonts) have similar vector representations. The resulting space can be used, with a reference font, to measure the visual similarity across different characters. Examples of characters and their nearest neighbors in homoglyph space are shown in Figure 9.

Using the cosine distance between characters in the homoglyph space as the substitution cost within a Levenshtein edit distance framework (Levenshtein et al., 1966) significantly improves linkage of firms and placenames. The study focuses on CJK, as the extremely large number of characters in this script make it completely infeasible to compute homoglyphs by hand, but shows that the method is extensible by computing homoglyphs for ancient Chinese characters and for all of Unicode. The broader takeaway is even when traditional methods—like string distance—are preferred, deep learning may provide a way to cheaply extend the methods to novel settings.



| language | Example Character | Character Inner Product Similarity Rank | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| Japanese | 畑 | 沺 | 㹨 | 佃 | 畋 | 細 |
| | 榊 | 鰰 | 𤝔 | 柿 | 柟 | 襧 |
| | 腺 | 原 | 泉 | 朖 | 湶 | 艆 |
| Korean | 냔 | 냐 | 년 | 논 | 는 | 샨 |
| | 농 | 능 | ㅎ | 동 | 등 | 닝 |
| | 닥 | 탁 | 박 | 댁 | 닷 | 학 |
| Traditional Chinese | 茶 | 荼 | 蒤 | 拳 | 荃 | 蒢 |
| | 梅 | 桉 | 侮 | 晦 | 毐 | 嗨 |
| | 蕻 | 萁 | 蘋 | 蔚 | 蒔 | 葑 |
| Simplified Chinese | 职 | 聍 | 聒 | 聆 | 瞑 | 取 |
| | 欢 | 炊 | 饮 | 软 | 钦 | 钐 |
| | 钢 | 纲 | 网 | 鋼 | 冈 | 惘 |

FIGURE 9. CHARACTER SIMILARITY, AS MEASURED BY VISION TRANSFORMERS.

### 3. *Linking unstructured data*

There is also a large NLP literature on linking entities mentioned in unstructured texts (*e.g.,* news, social media, etc.), a task referred to as *entity disambiguation*. Linking entity mentions in raw texts (tagged through NER - Section VI.4) to Wikipedia or other knowledge bases is useful, because these contain information such as structured biographical data. Whether individuals are in an external knowledge base may itself also be of interest.

Researchers might also wish to coreference entity mentions across documents in a corpus (*e.g.,* find every reference to President John F. Kennedy in historical news). This is referred to as *coreference resolution.*

Despite increasing digitization, historical documents typically lack cross-document identifiers for individuals mentioned in the texts, as well as identifiers from external knowledge bases like Wikipedia, both of which would make it much easier for economists to extract structured data from these sources.

Arora et al. (2024) develop a bi-encoder embedding model for coreferencing entities within texts and disambiguating them to Wikipedia. The model is contrastively trained on over 190 million entity pairs from Wikipedia. Positive pairs come from contexts (paragraphs) in Wikipedia that contain hyperlinks to the same page (for coreference), or from a context and the first paragraph of the relevant entity that it links to (for disambiguation). Hard negatives are mined at scale from Wikipedia disambiguation pages, which list entities that have confusable names or aliases. For example, the disambiguation page ”John Kennedy” includes John F. Kennedy the president, John Kennedy (Louisiana politician), John F. Kennedy Jr, and a variety of other John Kennedys. Hard negatives sample contexts mentioning John F. Kennedy (*e.g.,* with hyperlinks to John F. Kennedy's page) and pair them with contexts mentioning other entities from the John Kennedy disambiguation page. Hard negatives from families (*e.g.*, Henry Ford Jr. and Sr.) are over-represented by mining family members from Wikidata. It was also necessary to include random negatives, as otherwise the model lost its initial ability to distinguish easy cases, a phenomenon known in the deep learning literature as *catastrophic forgetting*.

This illustrates how existing knowledge can be mined to create informative negatives for contrastive training. More generally, Wikipedia is a useful source of training data (*e.g.,* firm aliases to train `LinkTransformer` models were also taken from Wikidata).

Entity mentions are disambiguated by embedding their contexts with the disambiguation model and retrieving their nearest Wikipedia neighbor in embedding space. If they are below a threshold cosine similarity to the nearest Wikipedia embedding, they are marked as not in the knowledge base. Figure 10 shows that entity mentions in 100 years of newswire coverage, spanning 1878-1977, correlate strongly with Wikipedia Qrank, which ranks Wikidata entities by aggregate page views in the Wikiverse Arora et al. (2024). Silcock et al. (2024) find that the most mentioned entity in these 100 years is Dwight Eisenhower—edging out Adolf Hitler, Richard Nixon, and Harry Truman—and only 4.7% of disambiguated entities in newswire articles are women.

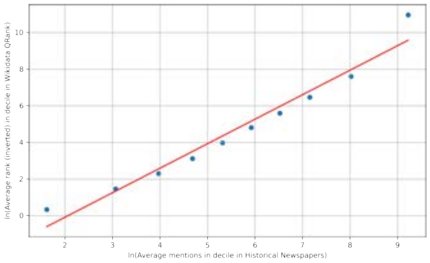


FIGURE 10. MENTIONS AGAINST WIKIPEDIA QRANK.

### 4. *Classification when categories are unknown*

The classes that a researcher would like to impute from unstructured data may be unknown ex ante. This is particularly likely when the aim is to describe the stylized facts in a novel unstructured corpus, but can arise more generally. This section provides several examples drawn from media economics: detecting reproduced article texts and images, classifying the biggest news stories historically, and retrieving historical news stories that are semantically similar to modern ones.

Reproduced content is a fundamental feature of media—both traditional media and in the age of sharing via social media. Media historian Julia Guarneri (2017) writes: “by the 1910s and 1920s, most of the articles that Americans read in their local papers had either been bought or sold on the national news market... This constructed a broadly understood

American 'way of life' that would become a touchstone of U.S. domestic politics and international relations throughout the twentieth century." Suppose we would like to be able to identify each unique article and image that was sent out over the newswire, measure how widely reproduced it was, and observe which papers reproduced it. This problem is more challenging than it seems. Texts are often heavily abridged and can contain significant OCR errors. Images are often cropped, and the quality can be extremely low.

Silcock et al. (2023) show that deep neural methods can significantly outperform non-neural methods for detecting noisily duplicated texts, with applications to historical newswires, modern news, and patent databases. Training and evaluation data are hand-curated by grouping articles across thousands of digitized local newspapers into groups of articles from the same newswire source. Articles from the same wire source are positives. Articles with high *n*-gram overlap but from different sources—often similar articles from different newswire services or article updates—form hard negatives. Random negatives are also used in training.

A bi-encoder embedding model is contrastively trained such that articles from the same wire article source (regardless of noise and abridgement) have similar vector representations, while articles from different sources (even if about the same underlying story) have different representations. These representations can then be clustered with highly efficient single-linkage clustering to quantify which articles are from the same underlying news wire or syndicate article source and which are from different ones. Community detection is used to break the spurious links that are a potential drawback of single-linkage clustering.[5] As with record linkage, the Sentence-BERT library (Reimers and Gurevych, 2019) was an important resource. The model is initialized with the S-BERT MPNet bi-encoder, a lightweight, high-performing semantic similarity model.

The neural approach outperforms traditional *N*-gram and hashing methods - sparse methods that rely on term overlap to detect noisy duplicates - by a wide margin. As suggested in Section VI.5, modest gains result from adding a re-ranking step that applies a cross-encoder to articles within a threshold bi-encoder distance.

A strength of embedding methods is their scalability. Clustering dense vector representations requires highly optimized similarity search, as traditional clustering libraries don't scale well. Facebook AI Similarity Search (FAISS) (Johnson, Douze and Jégou, 2019), an open-source library for computing vector similarity, made $10^{14}$ exact similarity comparisons—required to cluster 10 million article representations—on a single GPU card in around 3 hours. This could have been sped up significantly, with only a modest hit to accuracy, by using approximate vector search (with appropriately tuned hyperparameters).

Silcock et al. (2024) release 2.7 million unique newswire articles spanning 1878-1977 (the end date is due to copyright law changes). It includes topic tags, named entity tags, disambiguation of individuals to Wikipedia, and the counties where articles ran.

Detecting the noisy reproduction of images is analogous to detecting reproduced texts. Rather than training a language model, a vision model can be contrastively trained to map reproduced versions of the same image to similar vector representations and different images to dissimilar representations. A lightweight CNN works well in practice (Howard et al., 2019), with little gain from using a much larger ViT. Training data consist mostly of synthetically augmented images—which simulate the noise present in the actual images. When it is possible to simulate realistic synthetic data, this can save considerable annotation expense, though adding a modest number of labeled examples from target data may still offer a performance boost.

[5] Other common clustering methods, like hierarchical agglomerative clustering, do not scale well.

Embedding models are well-suited to assessing stylized facts in unstructured data at scale. Deep learning makes it possible to use a variety of novel unstructured datasets in economic research. While our focus is often on using causal estimation to test precisely defined hypotheses, understanding stylized facts is an important first step to formulating these hypotheses.

Consider an application from Dell et al. (2023), which constructs a historical newspaper dataset consisting of over 430 million historical U.S. newspaper articles. The aim of the exercise is to determine the biggest news stories of each year without knowing what these stories are ex ante. The study contrastively trains a model on data from AllSides, a modern news website that groups news articles from different sources into stories (often with different perspectives on the same event).[6] Grouped stories form positive pairs for training, with the model learning what constitutes the "same story" via these empirical examples. The trained model is used to embed articles, and stories are formed by clustering. Table 3 reports the largest cluster for each year (Dell et al., 2023). Some interesting stylized facts emerge, particularly the extensive coverage of labor movements. If a researcher wanted to create a measure of labor movements to use in a causal estimating equation, they would likely train a classifier with carefully crafted labels to predict which articles are about labor movements. In contrast, this exercise motivates why labor movements are important to study in the first place.

Franklin et al. (2024) use this model for additional data exploration, first masking out all named entities (people, organizations, locations, and other miscellaneous proper nouns) and then querying the most similar historical news articles to a modern news article query in embedding space. The resulting `News Déjà Vu` open-source package and website provide a novel tool for exploring parallels in how people have perceived past and present.

| **Year** | **Biggest story** |
|---|---|
| 1885 | Death of General Grant |
| 1886 | Southwest Railroad Strike |
| 1887 | Vatican supports Knights of Labor |
| 1888 | Rail strikes |
| 1889 | Samoan Crisis |
| 1890 | 1893 World's Fair planning |
| 1891 | New Orleans Lynchings |
| 1892 | Homestead Steel Strike |
| 1893 | World's Fair, Chicago |
| 1894 | Wilson–Gorman Tariff Act |
| 1895 | British occupation of Nicaragua |
| 1896 | Bimetallism Movement |
| 1897 | Coal Miners' Strike |
| 1898 | Cuban War of Independence |
| 1899 | Philippine-American War |
| 1900 | Anglo-Boer War |
| 1901 | U.S. Steel Recognition Strike |
| 1902 | Anthracite Coal Strike |
| 1903 | Panama Canal Treaty |
| 1904 | Russo-Japanese War |
| 1905 | Russo-Japanese Peace Process |
| 1906 | Hepburn Railroad Rate Bill |
| 1907 | Mining accidents |
| 1908 | Taft presidential victory |
| 1909 | Race to the North Pole |
| 1910 | Rail strikes |
| 1911 | Canadian Reciprocity Bill |
| 1912 | Republican National Convention |
| 1913 | Underwood-Simmons Tariff Act |
| 1914 | World War I |
| 1915 | World War I |
| 1916 | Pancho Villa Expedition |
| 1917 | World War I |
| 1918 | World War I |
| 1919 | Treaty of Versailles |
| 1920 | Rail strikes |

TABLE 3—BIGGEST NEWS STORIES.

### 5. *Optical character recognition*

Optical character recognition (OCR) is an important task for economists, particularly

[6] `https://www.allsides.com/unbiased-balanced-news`

for economic historians. Documents are extremely diverse in terms of character sets, languages, fonts or handwriting, printing technologies, and artifacts from scanning and aging. Off-the-shelf OCR technology is developed largely for small-scale commercial applications in high-resource languages like English, and the architecture it uses is not well-suited to extending OCR to lower-resource languages and settings, as elaborated in Section IX. OCR quality can deteriorate rapidly when moving away from English and a few other high-resource languages (Carlson, Bryan and Dell, 2024; Hegghammer, 2021). For example, on printed Japanese documents from the 1950s, the best-performing existing OCR mis-predicts over half of the characters. Poor performance is widespread, spurring a large post-OCR error-correction literature (Lyu et al., 2021; Nguyen et al., 2021; van Strien. et al., 2020).

Even in the highest resource settings, off-the-shelf solutions can still fail, especially when accuracy is paramount. This is particularly true when transcribing quantitative data. An OCR error in prose is often straightforward to correct in post-processing or inconsequential. However, with numbers, a similar error (*e.g.,* hallucinating a “1” at the beginning of a number) may severely bias downstream statistical analyses.

Moreover, the scale of document collections to be digitized can be vast. For example, the U.S. National Archives holds approximately 13.28 billion pages of textual records. Bringing big data to economic history requires an OCR technology that is both accurate and cheap to deploy.

If economists, historians, and others rely solely on off-the-shelf commercial technologies, we will end up focusing on economic applications that look a lot like high resource commercial applications (*e.g.,* receipts in English). This is indeed what I’ve seen over many years of working with students: they are far more likely to abandon projects in lower resource languages because the OCR quality of any existing off-the-shelf solution is poor. This skews economic knowledge towards settings that look more like high-resource commercial applications, which are not representative of the diversity of human societies.

To address these challenges, Carlson, Bryan and Dell (2024) develop a novel, open-source OCR architecture, `EffOCR` (**Eff**icient**OCR**). `EffOCR` is designed for researchers and archives seeking a sample-efficient, customizable, scalable OCR solution for diverse documents. Deep learning-based object detection methods (Section VIII) are used to localize individual characters or words in a document image. Recognition models for characters or words are contrastively trained—largely on augmented digital fonts—to map image crops of the same character or word to similar vector representations, regardless of font and other variations. Different characters or words, even if they have a very similar visual appearance, are mapped further apart.

A document is transcribed by embedding word or character crops and retrieving their nearest neighbors in an index that embeds crops rendered with a digital font. New characters or words can be added to the index after training (unlike with a classifier), a useful feature for economic historians since idiosyncratic symbols frequently appear in historical document collections.

`EffOCR` performs very accurately, even when using lightweight models designed for mobile phones that are cheap to train and deploy. For example, it can provide a sample efficient, highly accurate OCR architecture for historical Japanese documents where all current solutions fail. Its blend of accuracy and efficient runtime also makes it attractive for digitizing massive-scale collections in high-resource languages. Dell et al. (2023) cheaply digitize over 430 million historical newspaper articles from the Library of Congress’s Chronicling America collection with `EffOCR`. TrOCR, an open-source solution with similar accuracy, would have cost nearly 50 times more to deploy, and commercial so-

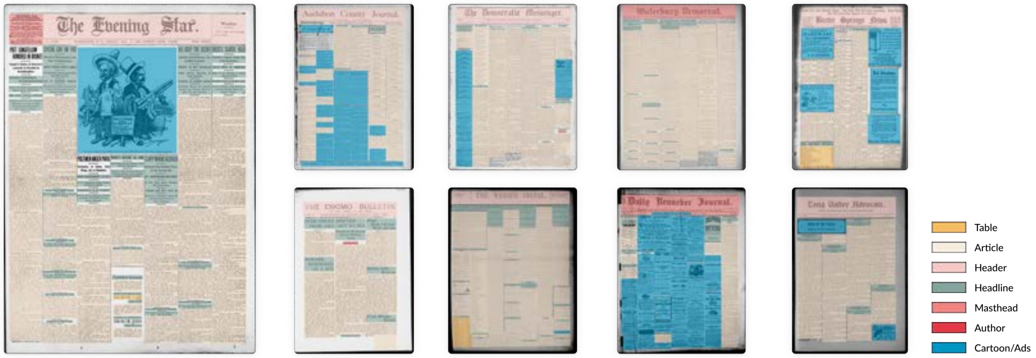


FIGURE 11. LAYOUTS FROM HISTORICAL NEWSPAPERS.

lutions were even more cost prohibitive.

EconDL links to a demo notebook for training a custom OCR for polytonic (ancient) Greek, using minimal cloud compute. Carlson, Bryan and Dell (2024) show that this model outperforms Google Cloud Vision on the target data. The notebook uses the `EffOCR` package (Bryan et al., 2023), which allows users to tune their own OCR models and run existing models off-the-shelf. `EffOCR` does not focus on handwriting; however, the approach would be analogous. Synthetic handwriting generators, *e.g.,* Bhunia et al. (2021), could provide extensive data for pre-training, analogous to the use of digital fonts.

## *VIII. Regression*

Regression is analogous to classification, except that a regression layer added to a neural network predicts a continuous number(s), rather than a set of class scores. For this reason, our treatment here is brief, focusing on a single application: object detection.

Object detection problems, as the name suggests, locate objects in an image (Ren et al., 2017; He et al., 2017; Kirillov et al., 2019; Cai and Vasconcelos, 2019; Redmon et al., 2016; Ultralytics, 2020; Carion et al., 2020; Liu et al., 2021). For example, an economist digitizing firm financial records would need to detect the coordinates of different document objects: *e.g.,* table headers, column and row headers, table cells, footnotes, etc. Alternatively, an economist wishing to measure informality from street view data would need to localize street vendors in an image. For each object, the neural network outputs four continuous numbers (top-x, top-y, height, and width of the box containing the object)—a regression problem—as well as the class of that object (*e.g.,* table header, column header, etc.)—a classification problem.

Figure 11 shows how object detection methods can be used to localize and classify document layout objects (*e.g.*, articles, headlines, etc.) in scans of historical newspapers, facilitating the creation of structured digital texts that can be analyzed with modern NLP methods. In contrast, Figure 12 provides an example of how commercial OCR (Google Cloud Vision) reads a newspaper scan like a single column book, failing to detect individual articles, headlines, etc. All one can do with these scrambled texts is search for keywords, as is typical in the economic literature using historical newspapers (see Hanlon and Beach (2022) for a review). Layout detection is also needed to extract structure when digitizing tabular data, as Figure 13 shows for historical Japanese firm records.

At present, document layout detection typically requires customization. An off-the-shelf

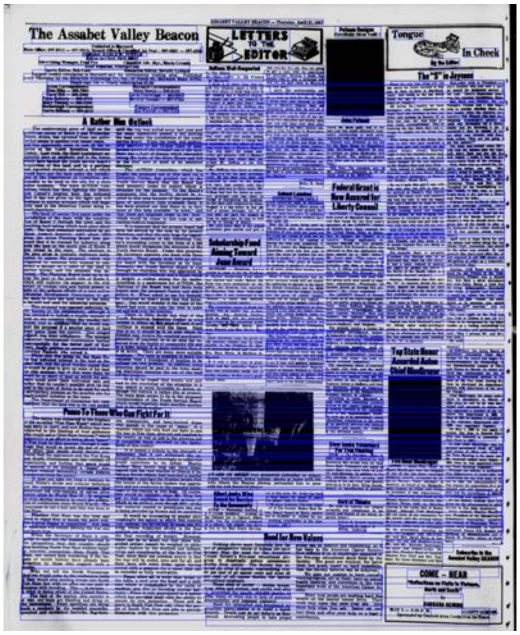

The Assabet Valley Beacon
LETTERS TO THE EDITOR
Tongue
In Cheek
Federal Grant is Now Assured for Liberty Council
Scholarship Fund

FIGURE 12. LAYOUTS AS DETECTED BY GOOGLE CLOUD VISION.

model may work well if the target task is quite close to what it was fine-tuned for. However, in computer vision, the main pre-training dataset is ImageNet, which consists of natural images, such as different breeds of dogs. Models have not been exposed to massive-scale pre-training on documents, and hence tend to suffer from substantial domain shift when applied to different types of documents. This is particularly true with historical documents, which are heterogeneous.

While there is a foundation model for object localization, Segment Anything by Meta AI (Kirillov et al., 2023), at the time of writing I have not found it to be very useful for document tasks. It can localize objects in an image, but does not classify them into categories. Moreover, localizations on document images are not particularly accurate at present.

The open-source package `Layout Parser` (Shen et al., 2021) lowers the barriers to detecting document layouts with deep learning.

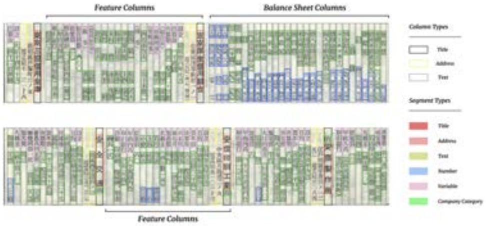


FIGURE 13. DOCUMENT LAYOUTS.

The library—which contains a model zoo for off-the-shelf usage and facilitates tuning customized models—is implemented with simple Python APIs. More information can be found on the EconDL resource page. The EconDL knowledge base introduces active learning for object detection (Shen et al., 2022), which chooses instances to label that the model is most uncertain about to economize on labeling costs.

Space constraints preclude delving into the architectures of object detection models, but a detailed treatment is provided in the object detection post in the EconDL knowledge base. Resources for satellite images include *e.g.,* Aleissaee et al. (2022); Wang et al. (2022); Bandara and Patel (2022); Fuller, Millard and Green (2022) and are linked from the page on processing satellite imagery in EconDL.

## *IX. Alternative methods*

This review has focused on a set of methods that—while a mainstay in deep learning—are far from comprehensive. If the reader delves into the deep learning literature, they will find other approaches to the above problems and may wonder why those were not covered. I focus on classifiers and embedding models because they are often sample and computationally efficient, meaning that they learn well from limited data and can be cheaply deployed on constrained hardware. They are user-friendly to train and can attain state-of-the-art performance on a variety of tasks.

This section provides a brief flavor of other ways of approaching two applications, OCR

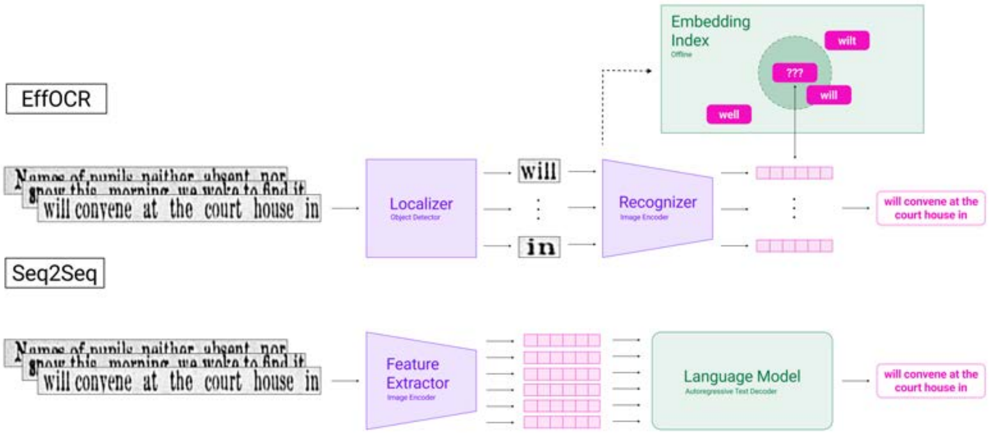

EffOCR
Seq2Seq
Names of pupils neither absent nor
snow this morning we woke to find it
will convene at the court house in
Localizer
Object Detector
will
in
Recognizer
Image Encoder
Embedding
Index
Offline
wilt
???
will
well
will convene at the court house in
Feature
Extractor
Image Encoder
Language Model
Autoregressive Text Decoder
will convene at the court house in

FIGURE 14. EFFOCR AND SEQ2SEQ MODEL ARCHITECTURES.

and entity disambiguation. It highlights how problems can be conceptualized differently and underscores some of the advantages of embedding models for academic applications.

### 1. *Optical Character Recognition*

Section VII.5 framed OCR as an image retrieval problem, using a contrastively trained vision model. This diverges from the literature, which mostly models OCR as a sequence-to-sequence (seq2seq) problem. Seq2seq models transform one sequence of data into another sequence. They are frequently used when the input and output data are sequences that may differ in length, such as in machine translation.

Figure 14 highlights the differences between the `EffOCR` and seq2seq architectures for OCR. First, seq2seq OCR typically requires line-level inputs and does not localize individual characters or words. Instead, it divides text line images or their representations into fixed-size patches. In contrast, `EffOCR` uses modern object detection methods (Cai and Vasconcelos, 2018; Jocher, 2020) to locate characters or words in the input image.

Second, seq2seq decodes image representations—created by a learned vision model—into text sequentially using a learned language model. Conversely, `EffOCR` employs contrastive training (Khosla et al., 2020) to learn a meaningful metric space for OCR. Its vision model projects crops of the same character or word close together, regardless of style, while projecting crops of different characters or words to dissimilar embeddings. With `EffOCR`, the vision embeddings alone are sufficient to infer text, by retrieving their nearest neighbor from an index created by embedding a digital font. In contrast, with seq2seq, the vision representations are decoded into texts with a language model, which requires jointly estimating millions of additional parameters.

A drawback of the seq2seq architecture is that it is challenging to extend and customize to novel settings (Hedderich et al., 2021), because training a joint vision-language model requires a vast collection of labeled image-text pairs and significant compute, particularly when state-of-the-art architectures are used. TrOCR—a transformer seq2seq model created by researchers at Microsoft—was trained using 684 million English synthetic text lines and 32 32GB V100 GPUs, a very costly setup that no academic researcher can replicate.

This drawback can be quantified by sample efficiency, which refers to how well a model can perform following exposure to a limited number of training examples. Some architectures learn more efficiently than others. Bi-encoders tend to learn efficiently, which is important for economists because our compute and annotation budgets are severely limited by deep learning standards.

Figure 15, drawn from Carlson, Bryan and Dell (2024), examines sample efficiency by training various open-source OCR architectures on the same small training sets. The x-axis plots the percentage of the `EffOCR` training dataset used in training, and the y-axis plots the character error rate.

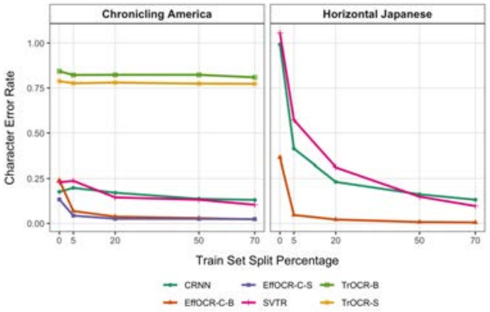


FIGURE 15. SAMPLE EFFICIENCY.

On just 99 labeled table cells for Japanese tables and 21 labeled rows from U.S. newspapers in the Chronicling America collection, as well as digital fonts (the 5% train splits), `EffOCR`'s character error rate is around 4%, showing viable few-shot performance. Other architectures, trained on identical data, remain unusable. `EffOCR` performs nearly as well using 20% of the training data as using 70%, where it continues to outperform all other alternatives. TrOCR learns almost nothing from the amount of data we can expose it to (hence why Microsoft trained it on 684 million text lines). In contrast, `EffOCR` can be trained with a student account in the cloud, or even on a laptop. CRNN is a much lighter weight, older seq2seq architecture, that learns better with limited data but uses an LSTM rather than a transformer, leading to an accuracy hit when fully trained.

Embedding models can also offer a computational advantage over seq2seq models. `EffOCR` supports inference parallelization across characters, promoting faster inference, whereas seq2seq requires autoregressive decoding, which is slower. `EffOCR` runs approximately 50 times faster than TrOCR, the only open-source model comparable in accuracy once fully trained. We created the open-source American Stories dataset consisting of over 430 million historical news articles with a $60,000 budget using `EffOCR`, and could not have realistically generated any substantial additional funds, let alone 50 times more funds.

A sample and compute efficient architecture makes it possible to achieve high-quality transcriptions in diverse settings and on massive-scale document collections, bringing big data to a diversity of economic history applications. In theory, contextual understanding from the full sequence of representations could lead to better OCR. In practice, state-of-the-art transformer seq2seq models are expensive to train and deploy, and are not available for lower-resource languages, with advances mainly in a few languages. By moving away from seq2seq models, significant improvements in sample and computational efficiency can be achieved. In an academic setting, such advantages are particularly relevant, as our applications are highly diverse and our budgets are usually extremely constrained.

## 2. *Entity Disambiguation*

Entity disambiguation—linking entity mentions in unstructured texts to an external knowledgebase like Wikipedia—has led to the development of various architectures. These include a masked language model (LUKE, Yamada et al. (2022)) and a neural translation model (GENRE, De Cao et al. (2020))—which uses a sequence-to-sequence architecture to translate mentions into Wikipedia ids—as well as the bi-encoder embedding architecture that treats entity disambiguation as a nearest neighbor retrieval problem (Wu et al., 2019). The latter architecture is used by this article's application (Section VII.3).

The masked language model approach masks out entities and predicts their Wikipedia id using a classifier head. While it is the leader on some benchmarks, in practice it has major limitations. Language models predict masked tokens through classification. LUKE is restricted to the top 50K Wikipedia entries, due to computational constraints in calculating the softmax. Many of the top-50K entries are not people, and many people that appear in historical news or government documents are not amongst the top-50K. Additionally, it does not accommodate out-of-knowledge base entities and requires sparse entity priors to initialize the model. In many applications, not all individuals will be in a knowledge base, and the model needs to be able to predict this.

The neural translation model's sequence-to-sequence architecture is slow during inference, taking approximately 60 times longer to run than bi-encoder embedding models. Arora et al. (2024) also show that an embedding model achieves higher accuracy disambiguating historical texts. In short, again a sequence-to-sequence architecture is costly to run at scale and doesn't necessarily offer performance advantages.

## *X. Conclusion*

Deep learning provides powerful tools for processing unstructured data. On social science tasks ranging from text classification to record linkage, entity disambiguation, and tracing the spread of reproduced content, deep learning can outperform traditional sparse methods by a wide margin (often by 20 points of F1/accuracy or more) (Silcock et al., 2023; Dell et al., 2023; Arora and Dell, 2024; Arora et al., 2023, 2024).

Deep learning can facilitate novel analyses by providing tools to impute structured information from unstructured data on a massive scale. When working with lightweight, sample-efficient pre-trained models, training and deployment are quite affordable, even for datasets with millions or billions of observations. For some applications, deep learning also offers promise for processing data from low-resource settings, with the potential to make economic research more representative of the diversity of human societies.

Becoming familiar with deep learning methods entails significant startup costs. This article—along with the accompanying open-source packages, tutorials, and knowledge base—aims to significantly reduce entry barriers for economists who would like to use deep learning in their research.

## *REFERENCES*

**Abramitzky, Ran, Leah Boustan, Katherine Eriksson, James Feigenbaum, and Santiago Pérez.** 2021. "Automated linking of historical data." *Journal of Economic Literature*, 59(3): 865–918.

**Alammar, Jay.** 2018*a*. "The illustrated Bert, Elmo, and Co. (how NLP cracked transfer learning)." `https://jalammar.github.io/illustrated-bert/`.

**Alammar, Jay.** 2018*b*. "The illustrated Transformer." `https://jalammar.github.io/illustrated-transformer/`.

**Alammar, Jay.** 2019. "The illustrated GPT-2 (Visualizing Transformer language models)." `http://jalammar.github.io/illustrated-gpt2/`.

**Alammar, Jay.** 2020. "How GPT3 Works - Visualizations and Animations." `https://jalammar.github.io/how-gpt3-works-visualizations-animations/`.

**Aleissaee, Abdulaziz Amer, Amandeep Kumar, Rao Muhammad Anwer, Salman Khan, Hisham Cholakkal, Gui-Song Xia, et al.** 2022. "Transformers in remote sensing: A survey." *arXiv preprint arXiv:2209.01206*.

**Ali, Alaaeldin, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al.** 2021. "Xcit: Cross-covariance image transformers." *Advances in Neural Information Processing Systems*, 34: 20014–20027.

**Angelopoulos, Anastasios N., Stephen Bates, Clara Fannjiang, Michael I. Jordan, Tijana Zrnic, and Emmanuel J. Candès.** 2023. "Prediction-Powered Inference." *arXiv preprint arXiv:2301.09633*.

**Archives, U.S. National, and Records Administration.** 2023. "The Soundex Indexing System." Accessed: 11/10/2023.

**Arora, Abhishek, and Melissa Dell.** 2024. "LinkTransformer: A Unified Package for Record Linkage with Transformer Language Models." *Association of Computational Linguistics: Systems Demonstration Track*.

**Arora, Abhishek, Emily Silcock, Leander Heldring, and Melissa Dell.** 2024. "Contrastive Entity Coreference and Disambiguation for Historical Texts." *arXiv preprint arXiv:2406.15576*.

**Arora, Abhishek, Xinmei Yang, Shao Yu Jheng, and Melissa Dell.** 2023. "Linking representations with multimodal contrastive learning." *arXiv preprint arXiv:2304.03464*.

**Athey, Susan, and Guido W Imbens.** 2019. "Machine learning methods that economists should know about." *Annual Review of Economics*, 11: 685–725.

**Bailey, Martha J, Connor Cole, Morgan Henderson, and Catherine Massey.** 2020. "How well do automated linking methods perform? Lessons from US historical data." *Journal of Economic Literature*, 58(4): 997–1044.

**Bandara, Wele Gedara Chaminda, and Vishal M Patel.** 2022. "A transformer-based siamese network for change detection." 207–210, IEEE.

**Bengio, Yoshua, Patrice Simard, and Paolo Frasconi.** 1994. "Learning long-term dependencies with gradient descent is difficult." *IEEE transactions on neural networks*, 5(2): 157–166.

**Bhunia, Ankan Kumar, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, and Mubarak Shah.** 2021. "Handwriting transformers." 1086–1094.

**Binette, Olivier, and Rebecca C Steorts.** 2022. "(Almost) all of entity resolution." *Science Advances*, 8(12): eabi8021.

**Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al.** 2020. "Language models are few-shot learners." *Advances in Neural Information Processing Systems*, 33: 1877–1901.

**Bryan, Tom, Jacob Carlson, Abhishek Arora, and Melissa Dell.** 2023. "EfficientOCR: An Extensible, Open-Source Package for Efficiently Digitizing World Knowledge"." *Empirical Methods on Natural Language Processing (Systems Demonstrations Track)*.

**Cai, Zhaowei, and Nuno Vasconcelos.** 2018. "Cascade r-cnn: Delving into high quality object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6154–6162.

**Cai, Zhaowei, and Nuno Vasconcelos.** 2019. "Cascade R-CNN: high quality object detection and instance segmentation." *IEEE transactions on pattern analysis and machine intelligence*, 43(5): 1483–1498.

**Cao, Hongliu.** 2023. “Recent advances in universal text embeddings: A Comprehensive Review of Top-Performing Methods on the MTEB Benchmark.” Amadeus SAS France.

**Carion, Nicolas, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko.** 2020. “End-to-end object detection with transformers.” 213–229, Springer.

**Carlson, Jacob., Tom. Bryan, and Melissa. Dell.** 2024. “Efficient OCR for Building a Diverse Digital History.” *ACL Anthology*.

**Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin.** 2021. “Emerging properties in self-supervised vision transformers.” 9650–9660.

**Cattaneo, Matias D., Yingjie Feng, Filippo Palomba, and Rocio Titiunik.** 2022. “Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption.” *Journal of Econometrics*, 228(2): 260–279.

**Chen, Xinlei, Saining Xie, and Kaiming He.** 2021. “An empirical study of training self-supervised vision transformers.” 9640–9649.

**Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins.** 2018. “Double/debiased machine learning for treatment and structural parameters.” *The Econometrics Journal*, 21(1): C1–C68.

**Chernozhukov, Victor, Kaspar Wüthrich, and Yinchu Zhu.** 2021. “An Exact and Robust Conformal Inference Method for Counterfactual and Synthetic Controls.” *Journal of the American Statistical Association*, 116(536): 1849–1868.

**Chernozhukov, Victor, Whitney K. Newey, and Rahul Singh.** 2022. “Automatic Debiased Machine Learning of Causal and Structural Effects.” *Econometrics Journal*, 25(3): C1–C38.

**Chopra, Sumit, Raia Hadsell, and Yann LeCun.** 2005. “Learning a similarity metric discriminatively, with application to face verification.” Vol. 1, 539–546, IEEE.

**Cui, Peng, and Susan Athey.** 2022. “Stable learning establishes some common ground between causal inference and machine learning.” *Nature Machine Intelligence*, 4(2): 110–115.

**De Cao, Nicola, Gautier Izacard, Sebastian Riedel, and Fabio Petroni.** 2020. “Autoregressive entity retrieval.” *arXiv preprint arXiv:2010.00904*.

**Dell, Melissa, Jacob Carlson, Tom Bryan, Emily Silcock, Abhishek Arora, Zejiang Shen, Luca D’Amico-Wong, Quan Le, Pablo Querubin, and Leander Heldring.** 2023. “American Stories: A Large-Scale Structured Text Dataset of Historical US Newspapers.” *Advances in Neural Information and Processing Systems, Datasets and Benchmarks*.

**Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei.** 2009. “Imagenet: A large-scale hierarchical image database.” 248–255, IEEE.

**Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.** 2019. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” 4171–4186, Association for Computational Linguistics.

**Dodge, Jesse, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner.** 2021. “Documenting large webtext corpora: A case study on the colossal clean crawled corpus.” *arXiv preprint arXiv:2104.08758*.

**Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al.** 2020. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929*.

**Ethayarajh, Kawin.** 2019. "How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings." *arXiv preprint arXiv:1909.00512*.

**Falkner, Stefan, Aaron Klein, and Frank Hutter.** 2018. "BOHB: Robust and efficient hyperparameter optimization at scale." 1437–1446, PMLR.

**Fernández-Villaverde, Jesús.** 2024. "Deep Learning for Macroeconomists." *https://www.sas.upenn.edu/~jesusfv/teaching.html*, https://www.sas.upenn.edu/~jesusfv/teaching.html.

**Franklin, Brevin, Emily Silcock, Abhishek Arora, Tom Bryan, and Melissa Dell.** 2024. "News Deja Vu: Connecting Past and Present with Semantic Search." 99–112.

**Fuller, Anthony, Koreen Millard, and James R Green.** 2022. "Transfer Learning with Pretrained Remote Sensing Transformers." *arXiv preprint arXiv:2209.14969*.

**Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford.** 2021. "Datasheets for datasets." 1–14.

**Gentzkow, Matthew, Bryan Kelly, and Matt Taddy.** 2019. "Text as Data." *Journal of Economic Literature*, 57(3): 535–574.

**Gissin, Daniel, and Shai Shalev-Shwartz.** 2019. "Discriminative active learning." *arXiv preprint arXiv:1907.06347*.

**Goh, Gabriel.** 2017. "Why momentum really works." *Distill*, 2(4): e6.

**Gong, Yuan, Yu-An Chung, and James Glass.** 2021. "Ast: Audio spectrogram transformer." *arXiv preprint arXiv:2104.01778*.

**Goodfellow, Ian, Yoshua Bengio, and Aaron Courville.** 2016. *Deep learning.* MIT press.

**Greff, Klaus, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber.** 2016. "LSTM: A search space odyssey." *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232.

**Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al.** 2020. "Bootstrap your own latent-a new approach to self-supervised learning." *Advances in Neural Information Processing Systems*, 33: 21271–21284.

**Guarneri, Julia.** 2017. *Newsprint Metropolis.* University of Chicago Press.

**Gururangan, Suchin, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith.** 2020. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks." *arXiv preprint arXiv:2004.10964*.

**Hanlon, W Walker, and Brian Beach.** 2022. "Historical Newspaper Data: A Researcher's Guide and Toolkit."

**Hedderich, Michael A., Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow.** 2021. "A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios." 2545–2568. Online:Association for Computational Linguistics.

**Hegghammer, Thomas.** 2021. "OCR with Tesseract, Amazon Textract, and Google Document AI: A benchmarking experiment." *Journal of Computational Social Science*, 5(1): 861–882.

**He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick.** 2017. "Mask R-CNN." 2961–2969.

**He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.** 2015. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." 1026–1034.

**He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.** 2016. "Deep residual learning for image recognition." 770–778.

**He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick.** 2022. "Masked autoencoders are scalable vision learners." 16000–16009.

**He, Pengcheng, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen.** 2020. "Deberta: Decoding-enhanced bert with disentangled attention." *arXiv preprint arXiv:2006.03654*.

**Hermans, Alexander, Lucas Beyer, and Bastian Leibe.** 2017. "In defense of the triplet loss for person re-identification." *arXiv preprint arXiv:1703.07737*.

**Hochreiter, Sepp, and Jürgen Schmidhuber.** 1997. "Long short-term memory." *Neural computation*, 9(8): 1735–1780.

**Holland, Sarah, Ahmed Hosny, Sara Newman, Josh Joseph, and Kevin Chmielinski.** 2018. "The Dataset Nutrition Label: A Framework to Drive Higher Data Quality Standards." *FAT* '18*, 1–5. New York, NY, USA:Association for Computing Machinery.

**Howard, Andrew, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al.** 2019. "Searching for mobilenetv3." 1314–1324.

**Ioffe, Sergey, and Christian Szegedy.** 2015. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." 448–456, PMLR.

**Jocher, Glenn.** 2020. "YOLOv5 by Ultralytics."

**Johnson, Jeff, Matthijs Douze, and Hervé Jégou.** 2019. "Billion-scale similarity search with gpus." *IEEE Transactions on Big Data*, 7(3): 535–547.

**Karpathy, Andrej.** 2022. "The spelled-out intro to neural networks and backpropagation." https://www.youtube.com/watch?v=VMj-3S1tku0.

**Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih.** 2020. "Dense passage retrieval for open-domain question answering." *arXiv preprint arXiv:2004.04906*.

**Khattab, Omar, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia.** 2022. "Demonstrate-Search-Predict: Composing Retrieval and Language Models for Knowledge-Intensive NLP." *arXiv preprint arXiv:2212.14024*.

**Khosla, Prannay, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan.** 2020. "Supervised contrastive learning." *Advances in Neural Information Processing Systems*, 33: 18661–18673.

**Kingma, Diederik P, and Jimmy Ba.** 2014. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*.

**Kirillov, Alexander, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al.** 2023. "Segment anything." *arXiv preprint arXiv:2304.02643*.

**Kirillov, Alexander, Ross Girshick, Kaiming He, and Piotr Dollár.** 2019. "Panoptic feature pyramid networks." 6399–6408.

**Korinek, Anton.** 2023. "Generative AI for Economic Research: Use Cases and Implications for Economists." *Journal of Economic Literature*, 61(4): 1281–1317.

**Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton.** 2012. "ImageNet Classification with Deep Convolutional Neural Networks." Vol. 25. Curran Associates, Inc.

**Lan, Zhenzhong, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.** 2019. "Albert: A lite bert for self-supervised learning of language representations." *arXiv preprint arXiv:1909.11942*.

**LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton.** 2015. "Deep learning." *Nature*, 521(7553): 436–444.

**Lei, Lihua, and Emmanuel J. Candès.** 2020. "Conformal Inference of Counterfactuals and Individual Treatment Effects." *arXiv preprint arXiv:2006.06138*.

**Levenshtein, Vladimir I, et al.** 1966. "Binary codes capable of correcting deletions, insertions, and reversals." Vol. 10, 707–710, Soviet Union.

**Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar.** 2017. "Hyperband: A novel bandit-based approach to hyperparameter optimization." *The Journal of Machine Learning Research*, 18(1): 6765–6816.

**Liu, Pengfei, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig.** 2023. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." *ACM Computing Surveys*, 55(9): 1–35.

**Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.** 2019. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692*.

**Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo.** 2021. "Swin transformer: Hierarchical vision transformer using shifted windows." 10012–10022.

**Liu, Zhuang, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie.** 2022. "A convnet for the 2020s." 11976–11986.

**Lynn, Veronica, Jonathan K. Kummerfeld, and Rada Mihalcea.** 2020. "A Causal Framework for Uncovering the Effects of Descriptive Text on Decision Making." 5276–5294.

**Lyu, Lijun, Maria Koutraki, Martin Krickl, and Besnik Fetahu.** 2021. "Neural OCR Post-Hoc Correction of Historical Corpora." *Transactions of the Association for Computational Linguistics*, 9: 479–483.

**Mehrabi, Ninareh, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan.** 2021. "A Survey on Bias and Fairness in Machine Learning." *ACM Computing Surveys (CSUR)*, 54(6): 1–35.

**Mehta, Sachin, and Mohammad Rastegari.** 2021. "MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer." *arXiv preprint arXiv:2110.02178*.

**Merchant, Amil, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney.** 2020. “What Happens to BERT Embeddings during Finetuning?” *arXiv preprint arXiv:2004.14448*.

**Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.** 2013. “Distributed representations of words and phrases and their compositionality.” *Advances in Neural Information Processing Systems*, 26: 3111–3119.

**Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru.** 2019. “Model Cards for Model Reporting.” 220–229.

**MLCommons.** 2024. “Croissant: A Metadata Framework for ML-Ready Datasets.” `https://github.com/mlcommons/croissant`, Accessed: 2024-07-09.

**Nguyen, Dat Quoc, Thanh Vu, Afshin Rahimi, Mai Hoang Dao, Linh The Nguyen, and Long Doan.** 2020. “WNUT-2020 task 2: identification of informative COVID-19 english tweets.” *arXiv preprint arXiv:2010.08232*.

**Nguyen, Thi Tuyet Hai, Adam Jatowt, Mickael Coustaty, and Antoine Doucet.** 2021. “Survey of Post-OCR Processing Approaches.” *ACM Comput. Surv.*, 54(6).

**Nielsen, Michael A.** 2015. *Neural Networks and Deep learning.* Vol. 25, Determination press San Francisco, CA, USA.

**Olah, Christopher.** 2014. “Deep learning, NLP, and Representations.” *GitHub blog, posted on July*.

**Oord, Aaron van den, Yazhe Li, and Oriol Vinyals.** 2018. “Representation learning with contrastive predictive coding.” *arXiv preprint arXiv:1807.03748*.

**Pennington, Jeffrey, Richard Socher, and Christopher D Manning.** 2014. “Glove: Global vectors for word representation.” 1532–1543.

**Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al.** 2019. “Language models are unsupervised multitask learners.” *OpenAI blog*, 1(8): 9.

**Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al.** 2021. “Learning transferable visual models from natural language supervision.” 8748–8763, PMLR.

**Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu.** 2019. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” *arXiv e-prints*.

**Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu.** 2020. “Exploring the limits of transfer learning with a unified text-to-text transformer.” *The Journal of Machine Learning Research*, 21(1): 5485–5551.

**Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi.** 2016. “You only look once: Unified, real-time object detection.” 779–788.

**Reimers, Nils, and Iryna Gurevych.** 2019. “Sentence-bert: Sentence embeddings using siamese bert-networks.” *arXiv preprint arXiv:1908.10084*.

**Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun.** 2017. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *IEEE transac-*

*tions on pattern analysis and machine intelligence*, 39(6): 1137–1149.

**Robins, James M., Andrea Rotnitzky, and Lue Ping Zhao.** 1994. “Estimation of regression coefficients when some regressors are not always observed.” *Journal of the American Statistical Association*, 89(427): 846–866.

**Rush, Alexander M.** 2018. “The annotated transformer.” 52–60.

**Russell, Robert C.** 1918. “U.S. Patent No. US1261167A.” *U.S. Patent and Trademark Office*, https://patents.google.com/patent/US1261167A/en.

**Sanderson, Grant.** 2017. “Neural Networks.” https://www.3blue1brown.com/topics/neural-networks.

**Sanderson, Grant.** 2020. “Convolutions in Image Processing.” https://www.youtube.com/watch?v=8rrHTtUzyZA.

**Sang, Erik F, and Fien De Meulder.** 2003. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition.” *arXiv preprint cs/0306050*.

**Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf.** 2019*a*. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” *arXiv preprint arXiv:1910.01108*.

**Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf.** 2019*b*. “DistilRoBERTa: A distilled version of RoBERTa.” *https://github.com/huggingface/transformers*, Accessed: 2024-07-09.

**Santurkar, Shibani, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry.** 2018. “How does batch normalization help optimization?” *Advances in Neural Information Processing Systems*, 31.

**Shafer, Glenn, and Vladimir Vovk.** 2008. “A Tutorial on Conformal Prediction.” *Journal of Machine Learning Research*, 9: 371–421.

**Shen, Qinlan, and Carolyn Rose.** 2021. “What Sounds “Right” to Me? Experiential Factors in the Perception of Political Ideology.” 1762–1771, Association for Computational Linguistics.

**Shen, Zejiang, Jian Zhao, Yaoliang Yu, Weining Li, and Melissa Dell.** 2022. “Olala: object-level active learning based layout annotation.” *EMNLP Computational Social Science Workshop*.

**Shen, Zejiang, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li.** 2021. “LayoutParser: A unified toolkit for deep learning based document image analysis.” 131–146, Springer.

**Silcock, Emily, Abhishek Arora, Luca D’Amico-Wong, and Melissa Dell.** 2024. “Newswire: A Large-Scale Structured Database of a Century of Historical News.” *arXiv preprint arXiv:2406.09490*.

**Silcock, Emily, Luca D’Amico-Wong, Jinglin Yang, and Melissa Dell.** 2023. “Noise-Robust De-Duplication at Scale.” Vol. 332.

**Simonyan, Karen, and Andrew Zisserman.** 2014. “Very deep convolutional networks for large-scale image recognition.” *arXiv preprint arXiv:1409.1556*.

**Stevens, Eli, Luca Antiga, and Thomas Viehmann.** 2020. *Deep learning with PyTorch.* Manning Publications Company.

**Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich.** 2015. “Going deeper with convolutions.” 1–9.

**Touvron, Hugo, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou.** 2021. "Training data-efficient image transformers & distillation through attention." 10347–10357, PMLR.

**Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample.** 2023. "LLaMA: Open and Efficient Foundation Language Models." *arXiv preprint arXiv:2302.13971*.

**Ultralytics.** 2020. "Ultralytics/yolov5." https://github.com/ultralytics/yolov5.

**van Strien., Daniel, Kaspar Beelen., Mariona Coll Ardanuy., Kasra Hosseini., Barbara McGillivray., and Giovanni Colavizza.** 2020. "Assessing the Impact of OCR Quality on Downstream NLP Tasks." 484–496, INSTICC. SciTePress.

**Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.** 2017. "Attention is all you need." *Advances in Neural Information Processing Systems*, 30.

**Vitercik, Ellen.** 2023. "Machine Learning for Algorithm Design." *https://vitercik.github.io/ml4algs/calendar/*, `https://vitercik.github.io/ml4algs/calendar/`.

**Wang, Di, Jing Zhang, Bo Du, Gui-Song Xia, and Dacheng Tao.** 2022. "An empirical study of remote sensing pretraining." *IEEE Transactions on Geoscience and Remote Sensing*.

**Wang, Feng, and Huaping Liu.** 2021. "Understanding the behaviour of contrastive loss." 2495–2504.

**Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou.** 2022. "Chain of Thought Prompting Elicits Reasoning in Large Language Models." *arXiv preprint arXiv:2201.11903*.

**Wilkerson, John, E. Scott Adler, Bryan D. Jones, Frank R. Baumgartner, Guy Freedman, Sean M. Theriault, Alison Craig, Derek A. Epp, Cheyenne Lee, and Miranda E. Sullivan.** 2023. "Policy Agendas Project: Congressional Bills." `https://comparativeagendas.net/datasets_codebooks`.

**Wu, Ledell, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer.** 2019. "Scalable zero-shot entity linking with dense entity retrieval." *arXiv preprint arXiv:1911.03814*.

**Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He.** 2017. "Aggregated residual transformations for deep neural networks." 1492–1500.

**Yamada, Ikuya, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto.** 2022. "Global entity disambiguation with BERT." 3264–3271.

**Yang, Xinmei, Abhishek Arora, Shao Yu Jheng, and Melissa Dell.** 2023. "Quantifying Character Similarity with Vision Transformers." *Empirical Methods on Natural Language Processing*.

**Zrnic, Tijana, and Emmanuel J. Candès.** 2023. "Cross-Prediction-Powered Inference." *arXiv preprint arXiv:2309.16598*.

TABLE S-1—PROMPTS USED TO QUERY OPENAI GPT MODELS

| Topic | Prompt |
|---|---|
| advice | We would like to classify whether a text is from an advice column. Advice columns answer letters from a reader seeking advice, or give unsolicited advice to readers. Examples include Dear Abby, Ask Ann Landers, Dear Doctor, etc. Yes/No: |
| antitrust | We would like to classify whether a text is about antitrust action. An article that is about antitrust action covers business practices that stifle competition, or accusations of such practices. It might involve legal action, government regulation, the breaking up of monopolies, or any plans to do so. Yes/No: |
| bible | We would like to classify whether a text reproduces a short religious blurb, like a quote from the Bible, prayer for the day, spiritual thought for the day, without explanations, opinions, interpretations, or discourses. Long sermons, discourse, or longer texts quoting from the Bible won't count as this 'short religious blurb.' This will generally look like 'X for the day' - short and crisp with no explanations or opinions at all besides the small blurb. Yes/No: |
| civil rights movement | We would like to classify whether a text is about the Civil Rights movement. This includes articles referring to organizations and individuals that protested racism against Black Americans, and articles discussing racial discrimination in the government, social and educational inequality of African Americans, police brutality against African Americans, the use of federal power to protect civil rights, or segregation. Articles about protests and riots are on topic if they stemmed from conflicts over civil rights or occurred after the death of Martin Luther King Jr., but race riots and acts of violence involving African Americans should not be on topic if they do not refer to discrimination. Some articles may refer to states rights or express anger about rioting—these articles are on topic only if race or federal protection of civil rights is mentioned. Yes/No: |
| contraception | We would like to classify whether a text is about contraception. Abortion is not contraception. Yes/No: |

TABLE S-2—PROMPTS USED TO QUERY OPENAI GPT MODELS (CONTINUED)

| **Topic** | **Prompt** |
|---|---|
| crime | We would like to classify whether a text covers about crime. This includes reports of crimes and investigations, coverage of court proceedings, law enforcement, and discussions of crime prevention and community safety. Violations of international law are not considered crimes. Nor are actions that may be unethical but are not illegal. Articles about Watergate should not be classified as about crime. Yes/No: |
| horoscope | We would like to classify whether a text is a newspaper horoscope. Horoscopes are articles that make predictions about people's futures based on astrological signals or signs (sun signs for instance) and are written to entertain and predict future events. They can be of daily frequency, monthly or even yearly. They can also be making astrological predictions for celebrities and known personalities. Yes/No: |
| labor movement | We would like to classify whether a text covers any American labor movements. A text that is on topic may cover unions striking, advocating for workers' rights, lobbying the government, or speaking about the experience of working in their industry. It may also cover statements by employers that express anger at union action. Articles that express support for or criticize the labor movement should both count as on topic. Yes/No: |
| obituaries | We would like to classify whether a text is an obituary. An obituary is defined as an article which mentions a death of a person and provides their Biographical/Family information or Funeral information. Articles which just mention deaths of individuals in passing and do not summarize information about their life or their funeral services are not obituaries. Yes/No: |
| pesticide | We would like to classify whether an article is about pesticides. Texts are on topic if they mention any chemical attempt to deter pests (insects, other animals, or fungi) that were on a crop or plant. This does not include herbicides, killing insects not on crops, non-chemical bug killing, or anything else about insects. Yes/No: |
| polio vaccine | We would like to classify whether a text is about the polio vaccine. Yes/No: |

TABLE S-3—PROMPTS USED TO QUERY OPENAI GPT MODELS (CONTINUED)

| Topic | Prompt |
|---|---|
| politics | We would like to classify whether a text is a political news. Yes/No: |
| protests | We would like to classify whether a text is about protests. Yes/No: |
| Red Scare | We would like to classify whether a text is about the Red Scare or McCarthyism. Any text that reports concerns about Communist infiltration into the U.S. government or institutions will be on topic. In addition, any accusations of U.S. citizens display communist sympathies will be on topic. References to the Korean War or the expansion of Communism abroad are not on topic, though fears of espionage in the U.S. by foreign Communists agents are on topic. Articles casting doubt on the extent of Communist influences in the U.S. should also be on topic. Yes/No: |
| schedules | We would like to classify whether a text is a schedule. This includes TV/Game/Movie/Church and other schedules. Schedules list multiple events or programs on the same medium or by the same organization. They are generally lists of time stamps of events followed by no (or a one line) description about them. A church calendar is a schedule, but a summary of the Sunday service is not. Movie listings are a schedule but a special screening of a particular movie is not. Yes/No: |
| sports | We would like you to classify whether a text is about sports. This does not include fishing, hunting, or flying. Yes/No: |

TABLE S-4—PROMPTS USED TO QUERY OPENAI GPT MODELS (CONTINUED)

| **Topic** | **Prompt** |
|---|---|
| Vietnam War | We would like you to classify whether a text pertains to (any aspect of) the Vietnam War. Yes/No: |
| weather | We would like to classify texts that contain measurements about the weather. These include the weather forecast as well as texts summarizing what the weather was in the recent past (e.g., temperature, precipitation, visibility). This also includes articles about weather records, like the coldest/hottest days in a decade/year or coldest/hottest places. If the article contains weather measurements but also talks about the consequences of the weather in passing (e.g., damage, events cancelled), we will call it about weather measurements. However, if it just talks about the consequences of the weather (e.g., closures, accidents, damage) without giving the weather measurements, this does not count. As a rule of thumb, if there is precise weather measurement somewhere in the article- call it a weather measurement article. Yes/No: |
| World War I | We would like to classify whether a text is about World War I. Yes/No: |