university of groningen

Quantitative Text Analysis – Essex Summer School

New developments in supervised machine learning

dr. Martijn Schoonvelde

University of Groningen

Today's Class

- Overview: New approaches in supervised machine learning
  - Weak supervision: learn from noisy labels
  - Transfer learning: reuse pre-trained models
- Flash talks Ladislav, Luca, Thijs
- Lab session: estimating word embeddings

New developments in supervised machine learning (sml)

In the traditional machine learning paradigm, large amounts of labeled data are often needed to **map input features to output labels**.

- But obtaining precise labels is costly and requires significant effort. The added value of supervised machine learning algorithms can be unclear due to high start-up costs (Guan & Kouda, 2022).

Recent advancements in SML introduce 'cheap learning' techniques that aim to reduce data requirements without compromising (and perhaps even improving) performance.

Two paths to cheaper supervision

Weak supervision	Transfer Learning	Underlying ideas
Generate labels using heuristics, crowds, or rules	Use pretrained model, fine-tune on small labeled data	Automate labeling using weak but scalable sources vs. reusing general language knowledge learned elsewhere
Many noisy labels	Few high-quality labels	Trade precision for scale (weak supervision) vs. reuse to reduce data needs (transfer learning)
Cheaper labeling at scale	Cheaper training	Reduce cost at different stages in the research pipeline
Label data first, then train a model	Use existing model, adapt to new task	Weak supervision builds the dataset; transfer learning starts with a trained model

Both approaches aim to reduce reliance on expensive gold-standard annotations.

Intuition behind weak supervision

- Traditional supervised learning requires labeling data **manually**, one example at a time – expensive and slow.
- Weak supervision replaces this with faster, high-level labeling strategies (e.g., rules, lexicons, crowd signals).
- Each source may be noisy or imperfect, but allows you to:
  - Combine multiple sources (**labeling functions**)
  - Observe when they **agree or conflict**
  - Learn how much to **trust each source**
- This allows one to produce a large-scale training set **automatically** at lower cost

Key components of weak supervision

1. Labeling Functions (LFs): Rules or models that assign labels (e.g., based on keywords, sentiment, or patterns)
2. Label Matrix: Rows = data points; columns = outputs of different LFs

Example: 3 tweets, 2 labeling functions for detecting hate speech (LF1 = profanity; LF2 = negative sentiment)

Tweet	LF1	LF2
You're an idiot!	1	1
I love peace	0	0
This is bad	0	1

Model learns to combine noisy labels (votes) from LFs.

Example: Hate Speech Detection with Weak Supervision

Step 1: Write Labeling Functions (LFs)
Each LF is a rule or model that votes on whether a tweet is hate speech.

- **Profanity detection**: Check for offensive words
- **Hate speech dictionary**: Match known slurs or hate terms
- **Sentiment analysis**: Flag very negative tone
- **User history**: Flag users with prior hate speech
- **Pattern matching**: Identify phrases like "go back to..."
- **All-Caps detection**: Flag shouting (potential signal)

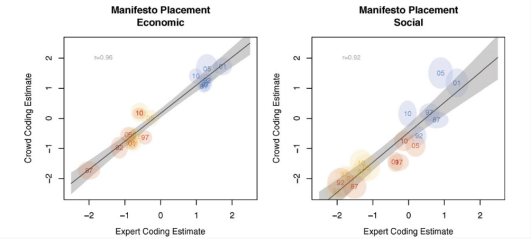
Step 2: Combine LF Outputs Resolve noisy and conflicting LF votes using a generative model.

- Infer **probabilistic labels** for each tweet
- Use these labels to **train a supervised classifier**

Crowdsourcing as weak supervision (Benoit et al. 2016)

- Show that non-experts can label text as reliably as experts
- Wisdom of the crowds: errors cancel out in aggregate
- Successful replication of CMP positions (at a fraction of the cost)

FIGURE 3. Expert and Crowd-sourced Estimates of Economic and Social Policy Positions



Drawbacks of traditional supervised learning

The key disadvantage of these classical machine learning algorithms is that they start the training process without any prior 'knowledge' of language or tasks (Laurer *et al. 2023*)

- Humans know that the words "attack" and "invasion" express similar meanings, or that the words "happy" and "not happy" tend to appear in different contexts

Conventional models on the other hand need to learn these language patterns and tasks from scratch with the training data as the only source of information.

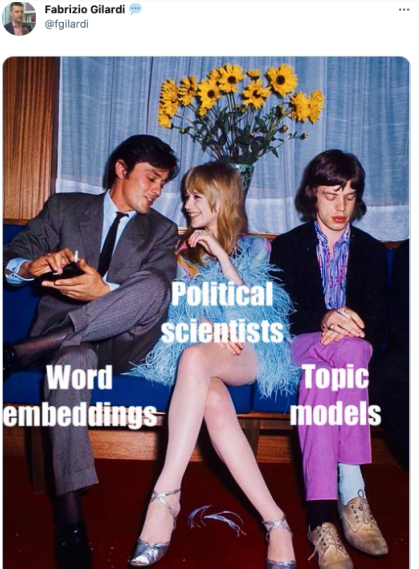
Developments in language representations: word embeddings

Word Embeddings (Mikolov et al., 2013):

- Represent words in similar contexts with similar vectors which can serve as input features for classifiers, providing "language knowledge."

Two core ideas (Van Atteveldt *et al.* 2022):

1. Meaning of a word can be expressed using a relatively small embedding vector, generally consisting of around 300 numbers which can be interpreted as dimensions of meaning.
2. These embedding vectors can be derived by scanning the context of each word in millions and millions of documents.



Developments in Language Representations: Word Embeddings

Table 1. Comparing Traditional Approaches with Embeddings

	Traditional Unsupervised	Traditional Supervised	Embeddings
Bag of words	Yes	Yes	No
Example models	Latent Dirichlet allocation; structural topic model	Support vector machines; random forest (RF)	GloVe, Word2Vec
Citations/applications	Quinn et al. (2010), Roberts et al. (2014)	Diermeier et al. (2012), Montgomery and Olivella (2018)	Rheault and Cochrane (2019), Rodman (2019)
Inputs	Document-term matrix	Document-term matrix; labeled y	Term-co-occurence matrix
Outputs	Document distribution over topics; topic distribution over words	Term importance matrix (for class prediction)	Word vectors
Example user decisions	Weighting of tokens; number of topics	Training/test split; weighting of tokens; number of trees (RF); number of variables at each split (RF); prior class probabilities	Pretrained or local fit; window size; embedding dimensions; weighting of tokens
Stability concerns	Multiple modes	Sensitivity to training/test set; labeling errors	Algorithmic; corpus characteristics

Source: Rodriguez & Spirling, 2022

Limitations of word embeddings

Laurer *et al.* (2023) argue that embeddings embed "shallow language knowledge."

- Each word gets a single vector, capturing one dominant meaning.

Consider, for example: "she put a date in his lunchbox"; "they went on a date"; "what's the date today?" (from: Kroon *et al.*, 2023)

- The embedding for date will be the same for all three uses of date. POS tagging doesn't help much since these are all nouns.

Transformers: core ideas

- Use self-attention to compare each word to every other word in a sentence
  - E.g., in "The cat sat on the mat," the model learns that "cat" relates to "sat"
- Represent each word in its full context (not just local neighbors)
  - Unlike older models (e.g., RNNs or bag-of-words)
- **Pre-train** on massive text (e.g., Wikipedia), then **fine-tune** on smaller, labeled datasets
  - E.g., fine-tune BERT for hate speech detection or sentiment classification

Popular models:

- **BERT** (Devlin et al., 2019): often used for text classification tasks
- **GPT**: often used for text generation tasks

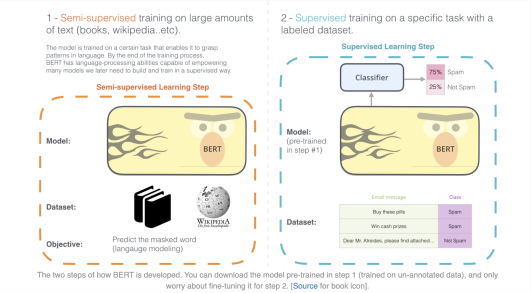
Transformer models

Relies on the **self-attention mechanism** (Vaswani *et al.*, 2017)

- For each word, the model asks: "Which other words in my context should I pay attention to?"
- Assigns attention weights to all other words, depending on relevance
- **Example:** In "She put a `date` in his `lunchbox`,"
  - The word "`date`" attends more to "`lunchbox`" than to "`calendar`"
  - This helps the model understand "`date`" refers to food, not time
- As a result, each word is represented in its context, not just as a static embedding
  - In essence, we obtain embeddings for each token, not for each type.

Pre-training transformer models

- LLM is pre-trained on a huge amount of unlabeled texts. At this stage the model acquires general language knowledge. In the case of BERT:
  1. **Masked language modeling (MLM) task**, which does not require manual annotation. Words are randomly hidden, and the model predicts the correct hidden words.
  2. **Next sentence prediction (NSP) task**: given two spans of text, the model predicts if these two spans appeared sequentially in the training corpus, outputting either [IsNext] or [NotNext]



The two steps of how BERT is developed. You can download the model pre-trained in step 1 (trained on un-annotated data), and only worry about fine-tuning it for step 2. [Source for book icon].

Source: https://jalammar.github.io/illustrated-transformer/

Transformer models in a nutshell

- **Understand words in full context**
  - The same word can have different meanings: ‘‘bank" of a river vs. ‘‘bank" loan
  - Transformers give each word a unique **contextualized embedding**
- **Handle long-range dependencies**
  - They can link related words even if they’re far apart in a sentence or paragraph
- **Efficient pretraining on massive datasets**
  - Trained on billions of words (e.g., Wikipedia, books, web)
  - Learn general language patterns that transfer across tasks

Transfer learning

A technique which involves taking a model trained in one context and applying it to another, making use of a minimal amount of retraining data.

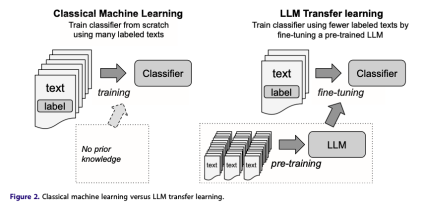


Figure 2. Classical machine learning versus LLM transfer learning.

Source: Kroon *et al.* 2023

1. Select a **pre-trained model**
2. **Fine-tune the model** to perform a classification task by providing it with labeled examples.
3. **Evaluate the model** using traditional metrics including precision, recall, F1, and accuracy

Because the pre-trained model already comes equipped with general language knowledge, the amount of data needed to fine-tune it is generally much lower compared to the amount of data needed to train a classic machine learning model.

Overview of sml approaches

Table 1. Overview of the advantages and disadvantages, domain-specificity, and generalizability of various automated techniques.

Method	Advantages	Disadvantages	Domain-Specificity	Generalizability
Dictionary-based classification	Easy to understand, computationally cheap, high degree of interpretability	Limited ability to capture semantic meaning.	Typically tailored to a specific domain	Generally limited performance of 'off-the-shelf' dictionaries to new domains.
Supervised machine learning based on Bag of Words representations	Generally improved performance over dictionary-based/rule-based classifiers, relatively computationally cheap	Limited ability to capture semantic meaning, performance is hindered by out-of-vocabulary words	Models tend to be domain-specific: Performance is best when training and test data are from the same domain.	Performance decreases in out-of-context domains.
Supervised machine learning based on static word embeddings	Better representation of semantic meaning and improved performance of out-of-vocabulary terms. If available, embedding models can easily be pre-trained on large amounts of unlabeled text data	The ability to capture context-dependent meaning is limited.	High domain specificity can be achieved if (1) word embeddings are of high quality and (2) trained on data representative of the application data	Word embeddings capture *static* semantic meaning that may be useful across domains, but generalizability is restricted due to a lack of contextual information.
Transfer learning (using contextualized embeddings from Large Language Models)	Capture context-dependent meaning, improving performance on unseen data.	Computationally expensive, limited interpretability, and the potential to encode structural and societal biases inherent in human communication.	When contextualized embeddings are trained on a data representative of the application domain (in terms of language and domain), domain-specificity tends to be high.	Potential for high generalizability as the pre-trained model can be fine-tuned on the domain-specific target data using *transfer learning*.

Choosing the right pre-trained model

- **Domain specificity** (e.g., biomedical BERT)
- **Language coverage** (monolingual vs. multilingual)
- **Model size and speed** (number of parameters, inference time)
- **Compute resources** (RAM, GPU requirements)

Resources:

- HuggingFace: model repository
- `text` and `grafzahl` packages (R + reticulate) ← **requires integration with Python**

Transfer learning on eight political science tasks (Laurer et al., 2023)

- 8 broad tasks (e.g., classifying topic, tone, populist style)
- Compared:
  - **BERT-NLI**, a fine-tuned model, against a traditional SML set-up (e.g., SVM with TF-IDF features)
- Key result:
  - Transfer learning with just 500 labeled examples matched or outperformed other SML models trained on 5,000 labels

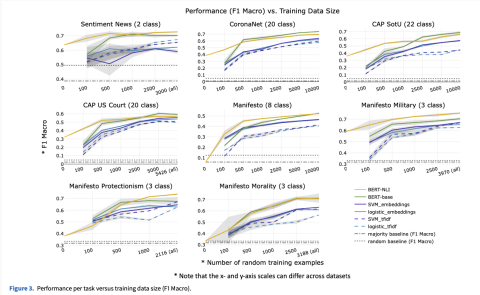


Figure 3. Performance per task versus training data size (F1 Macro).

Application: Emotion detection in political texts (Widmann & Wich (2022)

- Task: Detect emotions (e.g., **fear**, **anger**, **hope**) in political communication
- Compared:
  - **Transformer-based models** (Electra)
  - **Word embeddings**
  - **Domain-specific dictionaries** (ed8)
- Transformers performed considerably better across most emotional categories

Table 1. Precision, recall, and F1 scores for the three different approaches.

Emotions	Actual	Predicted	Precision	Recall	F1
ed8 dictionary					
Anger	508	281	0.83	0.46	0.59
Fear	189	287	0.43	0.66	0.52
Disgust	86	182	0.30	0.63	0.40
Sadness	201	289	0.41	0.59	0.48
Joy	143	179	0.46	0.58	0.52
Enthusiasm	220	248	0.44	0.50	0.47
Pride	158	247	0.31	0.48	0.38
Hope	305	303	0.53	0.53	0.53
Word-embeddings-based neural network approach					
Anger	508	500	0.80	0.78	0.79
Fear	189	152	0.61	0.49	0.55
Disgust	86	67	0.60	0.47	0.52
Sadness	201	122	0.70	0.42	0.53
Joy	143	92	0.68	0.44	0.54
Enthusiasm	220	176	0.64	0.51	0.57
Pride	158	123	0.52	0.41	0.46
Hope	305	265	0.69	0.60	0.64
Transformer-based (ELECTRA) approach					
Anger	508	495	0.85	0.83	0.84
Fear	189	221	0.60	0.70	0.64
Disgust	86	89	0.61	0.63	0.62
Sadness	201	181	0.64	0.57	0.60
Joy	143	122	0.70	0.59	0.64
Enthusiasm	220	242	0.62	0.68	0.65
Pride	158	151	0.61	0.58	0.60
Hope	305	352	0.68	0.78	0.73