

Neural Networks and Deep Learning

Lecture 6: NLP foundations: text representation and word embeddings





Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Turn financial text into numbers for natural language processing (NLP): tokens, vocabulary, bag of words and term frequency–inverse document frequency (TF-IDF).
- Write the objectives of word2vec and GloVe and train both on a real corpus of central-bank speeches.
- Build sentence classifiers from sparse and dense representations, with honest benchmarks and tests.
- Measure financial sentiment and topics on news and tweets, and know when a linear model is enough.

Abbreviations

AI	artificial intelligence
ASE	Bucharest University of Economic Studies
BERT	bidirectional encoder representations from transformers
BoW	bag of words
CBOW	continuous bag of words (word2vec variant)
CPU	central processing unit
ECB	European Central Bank
ES	expected shortfall
ESG	environmental, social and governance
FinBERT	BERT model adapted to financial text
IPE	Institute for Economic Forecasting, Romanian Academy
LLM	large language model
MLP	multilayer perceptron
NLP	natural language processing
OOV	out of vocabulary
PCA	principal component analysis
PMI	pointwise mutual information
SGNS	skip-gram with negative sampling
SVD	singular value decomposition
TF-IDF	term frequency–inverse document frequency
VaR	value-at-risk

The data: three real text collections

Corpus	Content	Size	Use here
Financial PhraseBank	sentences of financial news on Finnish listed companies, labelled positive / neutral / negative by 5–8 annotators	4 838 sentences	sentiment
Twitter financial news	finance tweets: bearish / bullish / neutral, and 20 topics	9 543 + 2 388 (sentiment), 16 990 + 4 117 (topic)	sentiment, topics
European Central Bank (ECB) speeches	full texts of the Executive Board and Supervisory Board	400 speeches, 1 289 896 tokens	our own embeddings

- Speeches from 8 Jan 2022 to 30 Oct 2026; PhraseBank by Malo et al. (2014); tweets as published on Hugging Face (train and validation splits).
- Every model is trained on the central processing unit (CPU) of a laptop with seed 20261108; test sets are used once.

(data/SOURCES.md)

Outline

- Text as data
- Sparse representations: bag of words and TF-IDF
- Dense representations: word embeddings
- Sentence classifiers: sparse versus dense
- Case study: topics of finance tweets

Tokenisation of a sentence

- **A real PhraseBank sentence**
  - "Finnish Aktia Group 's operating profit rose to EUR 17.5 mn in the first quarter of 2010 from EUR 8.2 mn in the first quarter of 2009 ."
  - Label given by the annotators: **positive**
- **Our tokeniser**
  - Lower case, words and numbers; every number becomes `<num>`; links and @-mentions removed
  - First tokens: `finnish aktia group s operating profit rose to eur <num> mn in the first` (27 in total)
- **Choices that change results**
  - Lower-casing merges "Bank" and "bank"; stemming merges "rose", "rising"
  - Removing stop words deletes "not", "down", "up": harmful for sentiment

- **In practice:** write the preprocessing down and keep it identical in training and production
  - A different tokeniser is a different model.

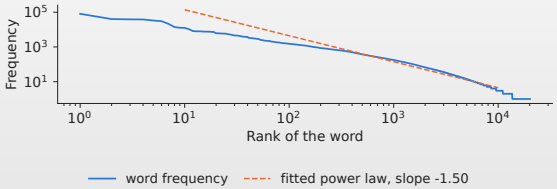
(Malo et al., 2014)

Experiment yourself: §1

Word frequencies and Zipf's law

- **In one sentence:** in 400 speeches of the European Central Bank, frequency falls as a power of rank.

$$f(r) \approx C\, r^{-a} \quad \Longleftrightarrow \quad \ln f(r) = \ln C - a \ln r$$



Source: European Central Bank, speeches dataset. Data: 8 Jan 2022 to 30 Oct 2026, 400 speeches.

- 1 289 896 tokens, 20 130 distinct words
  - Estimated slope -1.498 (ranks 10 to 10,000)
  - The 10 most frequent words make 25.0 % of all tokens.
- 32.5 % of the distinct words occur once: no model can learn them from this corpus alone.

(Zipf, 1949; own computation)

NN_L6_zipf

Outline

- Text as data
- Sparse representations: bag of words and TF-IDF
- Dense representations: word embeddings
- Sentence classifiers: sparse versus dense
- Case study: topics of finance tweets

Bag of words: a document is a vector of counts

- **In one sentence:** ignore the order, keep how many times each vocabulary word occurs.

$$\mathbf{x}_i = (c_{i1}, \ldots, c_{iV})^\top$$

$$c_{ij} = \#\{\text{occurrences of word } j \text{ in document } i\}$$

- **where**
  - V vocabulary size (words seen at least twice in the training part)
  - The document-term matrix $\mathbf{C} \in \mathbb{N}^{n \times V}$ is almost all zeros

- Bag of words (BoW) + multinomial logistic regression is the standard text baseline.
- n-grams add local order: "profit rose" and "profit fell" become features.
- Vocabulary is fitted on the training sentences only.

- **In practice:** BoW models are fast, cheap and explainable word by word: auditors like them.

(Salton & Buckley, 1988)

TF-IDF: lower weights for words that appear in every document

- **In one sentence:** a word is informative if it is frequent in this document and rare in the others.

$$w_{ij} = \underbrace{(1 + \ln c_{ij})}_{\text{tf}} \cdot \underbrace{\left(\ln \frac{1+n}{1+n_j} + 1\right)}_{\text{idf}}, \qquad \tilde{\mathbf{x}}_i = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|_2}$$

- **where**
  - $c_{ij} > 0$ count of term j in document i ($w_{ij} = 0$ if $c_{ij} = 0$)
  - n documents
  - n_j documents containing j
  - Sublinear tf and smoothed idf, as in scikit-learn

- Term frequency–inverse document frequency (TF-IDF) on unigrams and bigrams, then logistic regression with balanced class weights.
- Words like "the", "company", <num> get a low idf
  - "plunged" a high one.
- The ℓ_2 normalisation makes long and short sentences comparable.

- **In practice:** TF-IDF + logistic regression is the model any NLP vendor must beat on your own documents.

(Salton & Buckley, 1988)

The linear model learns a finance sentiment dictionary

- **In one sentence:** the difference between the positive and negative coefficients of each word, logistic regression on BoW, PhraseBank.

$$P(y = k \mid \mathbf{x}) = \frac{\exp(\beta_k^\top \mathbf{x} + \alpha_k)}{\sum_l \exp(\beta_l^\top \mathbf{x} + \alpha_l)}$$



Source: Financial PhraseBank (Malo et al., 2014). Data: 4,838 sentences of financial news, 5-8 annotators each.

- Most positive: rose, increase, grew, up, improved
  - Most negative: decreased, down, fell, cut, dropped.
- General-language dictionaries misread finance words ("liability", "tax"): learn the weights on finance text.

(Loughran & McDonald, 2011; own computation)

NN_L6_top_words

Outline

- Text as data
- Sparse representations: bag of words and TF-IDF
- Dense representations: word embeddings
- Sentence classifiers: sparse versus dense
- Case study: topics of finance tweets

The distributional hypothesis

- **In one sentence:** words used in similar contexts get similar vectors; similarity is the cosine of the angle.

$$\mathbf{e}_w = \mathbf{E}^\top \mathbf{o}_w \in \mathbb{R}^d, \qquad \cos(w, v) = \frac{\mathbf{e}_w^\top \mathbf{e}_v}{\|\mathbf{e}_w\| \, \|\mathbf{e}_v\|}$$

- **where**
  - $\mathbf{o}_w \in \{0, 1\}^V$ one-hot vector of word w
  - $\mathbf{E} \in \mathbb{R}^{V \times d}$ embedding matrix, $d = 100 \ll V = 8\,539$
  - A lookup, not a product, in code

- One-hot vectors are orthogonal: "inflation" and "prices" have cosine 0.
- In our own embeddings: cos(inflation, prices) = 0.49, cos(inflation, climate) = 0.16, cos(banks, lending) = 0.46.

- **In practice:** dense vectors let a model trained on "profit rose" generalise to "earnings increased".

(Firth, 1957; Bishop & Bishop, 2024, sec. 12.2)

word2vec skip-gram: predict the context from the word

- **In one sentence:** each word in a window around the centre word is predicted by a softmax over the vocabulary.

$$\max_{\mathbf{U},\mathbf{V}} \frac{1}{T}\sum_{t=1}^{T}\sum_{0<|j|\le c} \log p(w_{t+j} \mid w_t)$$

$$p(o \mid w) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_w)}{\sum_{k=1}^{V} \exp(\mathbf{u}_k^\top \mathbf{v}_w)}$$

- **where**
  - $\mathbf{v}_w$ input (centre) vector
  - $\mathbf{u}_o$ output (context) vector
  - c window radius
  - The denominator costs $O(V)$ per pair

- Continuous bag of words (CBOW) is the mirror image: predict the centre from the averaged context.
- The network is linear with one hidden layer
  - The embeddings are its first weight matrix.
- Our corpus: 3 679 754 (centre, context) pairs per epoch after subsampling.

- **In practice:** the full softmax over 8 539 words is too slow
  - Negative sampling replaces it.

(Mikolov et al., 2013)

Negative sampling: tell real pairs from random ones

- **In one sentence:** replace the softmax by $K+1$ binary logistic regressions: one real context word, K noise words.

$$\ell(w, o) = -\log \sigma(\mathbf{u}_o^\top \mathbf{v}_w) - \sum_{k=1}^{K} \log \sigma(-\mathbf{u}_{n_k}^\top \mathbf{v}_w)$$

$$n_k \sim P_n(w) \propto f(w)^{3/4}$$

- **where**
  - σ logistic function
  - $K = 5$
  - $f(w)$ corpus frequency
  - Frequent words kept with probability $\min(1, \sqrt{t/f(w)} + t/f(w))$, $t = 10^{-4}$

- Skip-gram with negative sampling (SGNS), $d = 100$, window 5, words with at least 5 occurrences, Adam, 5 epochs in 120 seconds.
- Loss per pair from 2.622 (epoch 1) to 2.223.
- SGNS implicitly factorises a shifted pointwise mutual information (PMI) matrix (appendix).

- **In practice:** the same trick, contrast a true item with sampled negatives, trains recommender and fraud-embedding systems.

(Mikolov et al., 2013; Levy & Goldberg, 2014)

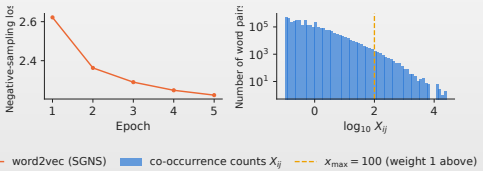
GloVe: fit the logarithm of the co-occurrence counts

- **In one sentence:** count once how often words appear together, then fit dot products to their logarithms with a weighted least-squares loss.

$$J = \sum_{i,j:\, X_{ij}>0} f(X_{ij}) \left(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \ln X_{ij} \right)^2$$

$$f(x) = \min\{1, (x/x_{\max})^\alpha\}$$

- **where**
  - X_{ij} co-occurrences within 10 words, weighted $1/\text{distance}$
  - $x_{\max} = 100$, $\alpha = 0.75$
  - Final vector $\mathbf{w}_i + \tilde{\mathbf{w}}_i$



Source: European Central Bank, speeches dataset. Data: 8 Jan 2022 to 30 Oct 2026, 400 speeches.

- **In practice:** 3 841 517 non-zero pairs (5.269 % of V^2)
  - 99.81 % are below $x_{\max}$ and get a weight below 1.

(Pennington et al., 2014; own computation)

NN_L6_emb_training

Count-based word vectors: PMI and truncated matrix factorisation

- **In one sentence:** before neural embeddings, dense vectors came from factorising a matrix of word associations.

$$\mathrm{PMI}(w, c) = \ln \frac{P(w, c)}{P(w)P(c)}$$

$$\mathbf{M}_+ \approx \mathbf{U}_d \boldsymbol{\Sigma}_d \mathbf{V}_d^\top, \qquad \mathbf{E} = \mathbf{U}_d \boldsymbol{\Sigma}_d^{1/2}$$

- **where**
  - $P(w, c)$ share of co-occurrences of w and c
  - $\mathbf{M}_+$ the matrix of $\max(\mathrm{PMI}, 0)$
  - $\mathbf{U}_d, \boldsymbol{\Sigma}_d, \mathbf{V}_d$ the d leading terms of the singular value decomposition (SVD)

- Pointwise mutual information (PMI) is positive when two words meet more often than chance.
- Our co-occurrence table has 3 841 517 non-zero pairs, 5.269 % of all V^2 cells: sparse, so the SVD is cheap.
- word2vec, GloVe and the SVD of PMI give similar neighbours on the same corpus.

- **In practice:** a factorisation of counts is transparent and reproducible: a good first embedding for an internal corpus.

(Levy & Goldberg, 2014)

Nearest neighbours in our central-bank embeddings

Word	word2vec (SGNS)	GloVe
`inflation`	headline, underlying, revert, hovering, medium	expectations, underlying, headline, core, term
`energy`	fertilisers, food, hicpxx, prices, renewable	prices, food, shock, commodity, renewable
`banks`	repaying, supervisors, themselves, obtaining, supervised	central, their, supervisors, they, have
`climate`	change, environmental, degradation, nature, ipcc	change, nature, environmental, related, risks
`digital`	convenient, functionality, kantar, erpb, payments	euro, payments, form, would, cash
`wages`	wage, profits, catch, negotiated, salaries	real, nominal, higher, incomes, rising
`ukraine`	invasion, russia, aggression, russian, brutal	invasion, russia, war, unjustified, russian
`rates`	interest, rate, compressed, hikes, repriced	interest, rate, short, lending, long

- Five nearest words by cosine; both trained on the same 1 289 896 tokens of ECB speeches; mean overlap of the two top-5 lists: 1.88 words.

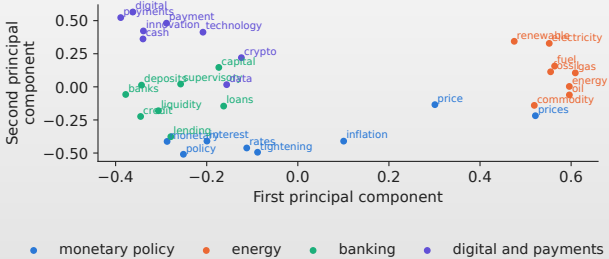
- **In practice:** a central-bank watcher can expand a search for "inflation" with its learned neighbours instead of a hand-made list.

(own computation)

Experiment yourself: §2

Topics of central banking form regions of the embedding space

- **In one sentence:** 2-D principal component analysis (PCA) projection of 32 words chosen in four groups; the groups were not given to the model.



Source: European Central Bank, speeches dataset. Data: 8 Jan 2022 to 30 Oct 2026, 400 speeches.

- Two components keep 31.9 % of the variance of these 32 vectors, so the picture is a rough shadow of $\mathbb{R}^{100}$.

(own computation)

NN_L6_w2v_pca

Analogies, and the limits of one vector per word

$a : b :: c : ?$	answer
germany : german : : france	fran
italy : italian : : spain	commissioned
banks : bank : : households	household
rise : rising : : fall	buffered
higher : lower : : increase	marginal

- Answer $= \arg\max_v \cos(\mathbf{e}_v, \mathbf{e}_b - \mathbf{e}_a + \mathbf{e}_c)$, excluding a, b, c; our word2vec, one small corpus.

(Mikolov et al., 2013; Bojanowski et al., 2017)

- **Static embeddings**
  - One vector per word type, whatever the sentence
  - "interest" (rate) and "interest" (attention) share a vector; so do "bank" and "bank"
- **Coverage**
  - The ECB vocabulary covers 84.7 % of PhraseBank tokens and 76.4 % of tweet tokens
  - The rest are out of vocabulary (OOV); subword models (fastText) and transformers (Lecture 7) address both problems

Subword vectors for words never seen

- **In one sentence:** build a word vector from the vectors of its character n-grams, so a new ticker or a typo still gets a vector.

$$\mathbf{e}_w = \sum_{g \in \mathcal{G}(w)} \mathbf{z}_g$$

- **where**
  - $\mathcal{G}(w)$ the character n-grams of w (for example 3 to 6 characters, with boundary symbols) and the word itself
  - $\mathbf{z}_g$ learned n-gram vectors

- Coverage of our word-level ECB vocabulary: 84.7 % of PhraseBank tokens, 76.4 % of tweet tokens.
- Subword models (fastText) share information between “profit”, “profits” and “profitability”.
- Transformers (Lecture 7) use the same idea with learned subword tokens.

- **In practice:** for tweets and chat messages, with tickers and spelling variants, use subword vectors or a subword tokeniser.

(Bojanowski et al., 2017)

Outline

- Text as data
- Sparse representations: bag of words and TF-IDF
- Dense representations: word embeddings
- Sentence classifiers: sparse versus dense
- Case study: topics of finance tweets

An embedding-bag network: averaged word vectors followed by a softmax layer

- **In one sentence:** the simplest neural text classifier learns the embeddings and the classifier together.

$$\mathbf{z} = \frac{1}{|S|} \sum_{s \in S} \mathbf{e}_s$$

$$\hat{\mathbf{p}} = \text{softmax}(\mathbf{W}\,\text{dropout}(\mathbf{z}) + \mathbf{b})$$

$$\mathcal{L} = -\frac{1}{n} \sum_i \omega_{y_i} \log \hat{p}_{i,y_i}$$

- Random start, or word2vec start: rows of $\mathbf{E}$ initialised with our ECB vectors, then fine-tuned.
- Also compared: mean word2vec vector + logistic regression (frozen), TF-IDF + a multilayer perceptron (MLP) with 128 hidden units.

- **where**
  - S the tokens and bigrams of the sentence
  - $\mathbf{e}_s$ learned vectors ($d = 100$)
  - $\omega_k = n/(3n_k)$ class weights against the dominance of "neutral"

- **In practice:** this is the architecture of fastText-type classifiers, used in production for millions of documents per hour.

(Joulin et al., 2017)

Logistic regression on text: the gradient

- **In one sentence:** the multinomial logistic model is a one-layer network; its gradient compares predicted and true class shares.

$$\mathcal{L}(\mathbf{B}) = -\sum_i \sum_k y_{ik} \ln \hat{p}_{ik} + \frac{1}{2C}\|\mathbf{B}\|_F^2$$
$$\nabla_{\mathbf{B}} \mathcal{L} = \mathbf{X}^\top (\hat{\mathbf{P}} - \mathbf{Y}) + \mathbf{B}/C$$

- **where**
  - $\mathbf{X}$ the TF-IDF matrix ($n \times V$, sparse)
  - $\mathbf{Y}$ one-hot labels
  - $\hat{\mathbf{P}}$ softmax probabilities
  - C inverse penalty, chosen on validation from $\{0.1, 1, 10\}$

- Convex loss: one optimum, no seed to report
  - The multilayer perceptron (MLP) on the same inputs adds a hidden layer of 128 units.
- PhraseBank macro-F1: logistic 73.49 %, MLP 71.37 %: the hidden layer does not help here.

- **In practice:** with a few thousand short sentences, the extra flexibility of a network buys variance, not accuracy.

(own computation)

Sentence vectors built from word vectors

- **In one sentence:** averaging word vectors is the simplest sentence representation, and it loses what matters for sentiment.

$$\mathbf{s}_i = \frac{1}{|S_i|} \sum_{w \in S_i} \mathbf{e}_w, \qquad \mathbf{s}_i^{\mathrm{w}} = \frac{\sum_w a_w \mathbf{e}_w}{\sum_w a_w}, \quad a_w = \frac{\alpha}{\alpha + f(w)}$$

- **where**
  - S_i the in-vocabulary tokens of sentence i
  - a_w a weight that down-weights frequent words
  - α a small constant

- Mean of our frozen ECB vectors + logistic regression: macro-F1 57.30 % on PhraseBank, 53.71 % on tweets.
- "rose" and "fell" are close in a co-occurrence space (same contexts), so the average blurs the sign.
- Fine-tuning the vectors on labelled data (embedding bag) recovers most of the gap.

- **In practice:** generic embeddings are good for search and similarity, not automatically for sentiment.

(own computation)

Protocol and metric: macro-F1 on an imbalanced target

- **In one sentence:** "neutral" is 59.3 % of PhraseBank, so accuracy rewards a model that says "neutral" all the time.

$$\mathrm{F1}_k = \frac{2P_kR_k}{P_k + R_k}, \qquad \text{macro-F1} = \frac{1}{3}\sum_{k=1}^{3} \mathrm{F1}_k$$

- **where**
  - P_k precision and R_k recall of class k
  - Each class counts equally, whatever its size

- PhraseBank: 5 random stratified splits 70/15/15
  - Tweets: published validation split used as test, 3 seeds.
- Penalty C, early stopping and vocabulary are chosen on the validation part only.

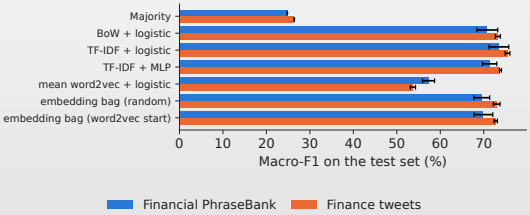
- **In practice:** for an alert system, report the recall of the negative class next to the macro-F1: that is the class that costs money.

(own computation)

Experiment yourself: §3

On short financial sentences, TF-IDF with logistic regression has the highest macro-F1

- **In one sentence:** test macro-F1 of seven representations and classifiers, PhraseBank and finance tweets (bars: mean, whiskers: standard deviation).



Source: Financial PhraseBank (Malo et al., 2014); Twitter financial news (Hugging Face, zeroshot). Data: PhraseBank: 5 random 70/15/15 splits; tweets: published validation split as test, 3 seeds.

- Best on PhraseBank: 73.49 (TF-IDF + logistic) vs 69.93 (embedding bag, word2vec start)
  - Majority 24.81.

(own computation)

NN_L6_sentiment_models

Sentiment results, with uncertainty

Model	PhraseBank macro-F1	acc.	Tweets macro-F1	acc.
Majority class	24.81 ± 0.00	59.28	26.40 ± 0.00	65.58
BoW + logistic	70.82 ± 2.41	74.94	73.25 ± 0.59	80.00
TF-IDF + logistic	73.49 ± 2.28	77.69	75.47 ± 0.56	81.77
TF-IDF + MLP	71.37 ± 1.65	76.23	73.85 ± 0.24	80.56
mean word2vec + logistic	57.30 ± 1.39	62.01	53.71 ± 0.60	61.59
embedding bag, random start	69.56 ± 1.81	74.11	72.97 ± 0.78	79.77
embedding bag, word2vec start	69.93 ± 2.16	74.99	72.76 ± 0.39	79.51

- Paired over the 5 PhraseBank splits: TF-IDF + logistic minus embedding bag (word2vec start) $= +3.55$ points, $t = 9.88$ (4 degrees of freedom).
- Starting from our ECB word2vec vectors changes the embedding bag by $+0.37$ points on PhraseBank and -0.21 on tweets.

- **In practice:** the cheapest model within one standard deviation of the best is the one to deploy and explain.

(own computation)

Choice of the embedding dimension

Dimension d	Macro-F1 (%)
8	67.3
16	66.5
32	67.2
64	67.8
128	68.8
256	67.3

- Embedding bag, random start, PhraseBank split 1, mean of 3 seeds.

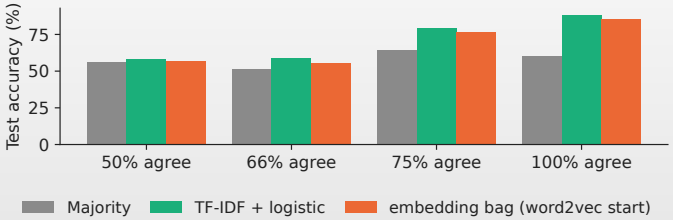
- Parameters of the embedding layer: $V \times d$
  - With bigrams V is several thousand, so d drives the model size.
- Here the score is flat in d: from 67.3 at $d = 8$ to 68.8 at $d = 128$.
- A three-class task on short sentences needs few dimensions
  - General-purpose embeddings use 100–300 because they serve many tasks.

- **In practice:** choose d on validation data
  - A smaller model is cheaper to serve and to explain.

(own computation)

Classification errors and annotator agreement

- **In one sentence:** test accuracy by the share of annotators who agreed on the label of the sentence.



Source: Financial PhraseBank (Malo et al., 2014). Data: test sentences of 5 random splits, by share of annotators that agree.

- TF-IDF + logistic: 88.4 % on unanimous sentences, 58.1 % on 50 % agreement
  - 2 259 of 4 838 sentences are unanimous.

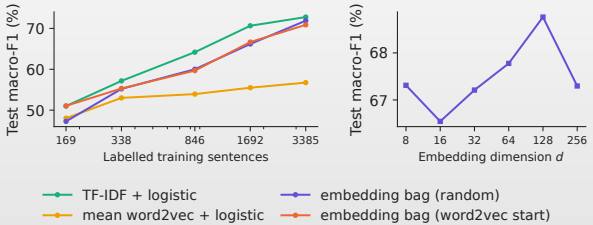
- **In practice:** label noise caps every model: before buying a bigger model, check how much your own analysts agree.

(own computation)

NN_L6_agreement

Learning curves with few labels: word2vec initialisation and TF-IDF

- **In one sentence:** left: test macro-F1 against the number of labelled sentences; right: embedding size of the embedding bag.



Source: Financial PhraseBank (Malo et al., 2014). Data: left: 3 random splits; right: embedding bag (random start), split 1, 3 seeds.

- With 169 labelled sentences: TF-IDF 51.0, mean word2vec 48.0, embedding bag random 47.2, word2vec start 51.0
  - With all 3 385: 72.8, 56.7, 71.9, 70.9.

(own computation)

NN_L6_learning_curve

Main error type: negative news classified as neutral

True \ predicted	neg.	neu.	pos.
negative	–	107	48
positive	50	263	–

- TF-IDF + logistic, the 5 PhraseBank test sets pooled (3 635 sentences).

- Recall: negative 65.9 %, neutral 84.1 %, positive 69.5 %.
- Opposite-sign errors are rare
  - Most errors move a sentence to neutral.
- Typical miss: a fall described with neutral words and numbers (“sales were EUR 10 mn, down from 12”).

- **In practice:** numbers carry sentiment in finance
  - Mapping them all to one token loses “down from”-type comparisons that a sequence model can read.

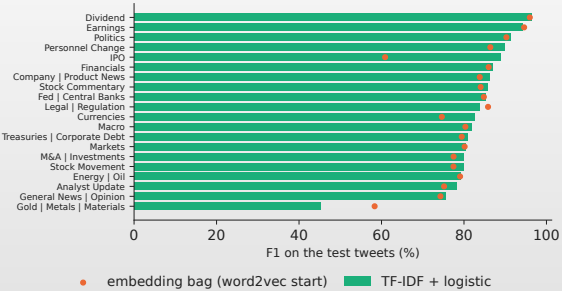
(own computation)

Outline

- Text as data
- Sparse representations: bag of words and TF-IDF
- Dense representations: word embeddings
- Sentence classifiers: sparse versus dense
- Case study: topics of finance tweets

Topic classification: F1 is high for earnings and the Fed, low for general news

- **In one sentence:** per-topic F1 on the 4 117 test tweets (published validation split).



Source: Twitter financial news datasets (Hugging Face, zeroshot). Data: 4,117 test tweets (published validation split), 20 topics.

- Macro-F1: TF-IDF + logistic 82.62 %, embedding bag (word2vec start) 80.45 %
  - Accuracy 84.48 % vs 20.69 % for the majority topic.

(own computation)

NN_L6_topic_f1

Applications of text models in finance

- **News and tweet triage**
  - Route items by topic (best: Dividend, F1 96.4 %) and flag negative ones for the analyst
- **Central-bank communication**
  - Track words near “inflation” or “energy” speech by speech
- **Compliance and environmental, social and governance (ESG) screening**
  - Search reports with learned synonyms, not fixed keyword lists
- **Limits of today’s models**
  - Word order beyond bigrams is lost
  - One vector per word ignores context
  - Weakest topic: Gold | Metals | Materials, F1 45.2 %
- **Next: attention**
  - Contextual vectors from transformers, e.g. bidirectional encoder representations from transformers (BERT) and a BERT model adapted to financial text (FinBERT), in Lecture 7, compared fairly with these baselines

- **In practice:** start every text project with TF-IDF + logistic regression
  - Move to embeddings or transformers only when the gain is shown on your data.

(own computation)

Quiz 1: idf

- A word appears in every one of the n training sentences
  - What is its smoothed idf?

Answer

- In $\frac{1+n}{1+n} + 1 = 1$: the minimum, so it barely moves the normalised vector.

Quiz 2: negative sampling

- Why does word2vec draw negative words from $f(w)^{3/4}$ and not from $f(w)$?

Answer

- The power 3/4 flattens the distribution: rare words are drawn more often than their frequency, frequent ones like "the" less, which gives more useful contrasts.

Quiz 3: accuracy

- A vendor reports 60 % accuracy on PhraseBank sentiment
  - Is it good?

Answer

- No: always "neutral" gives 59.28 %
  - Ask for macro-F1 (our TF-IDF + logistic: 73.49 %).

Quiz 4: transfer

- You have only 169 labelled sentences
  - Which representation helps most?

Answer

- Compare on validation data: here mean word2vec gives 48.0, TF-IDF 51.0, embedding bag with word2vec start 51.0 macro-F1.

Quiz 5: polysemy

- Why can a static embedding not separate "interest rate" from "of interest to investors"?

Answer

- It has one vector per word type, the average of all its uses
  - Contextual models (Lecture 7) give a different vector in each sentence.

Wrap-up

- Text as data
- Sparse representations: bag of words and TF-IDF
- Dense representations: word embeddings
- Sentence classifiers: sparse versus dense
- Case study: topics of finance tweets

- Text becomes numbers through tokens and counts (BoW, TF-IDF) or dense vectors (word2vec, GloVe) learned from co-occurrence.
- On real financial sentences TF-IDF + logistic regression is the model to beat
  - A word2vec start helps the network only when labels are few.
  - Next: attention and transformers (Lecture 7).

Research from our group: language models for finance

- **Large language models (LLM) forecast value-at-risk (VaR) and expected shortfall (ES)** (Pele, Bolovăneanu, Lin, Ren, Ginavar, Spilak, Andrei, Toma, Lessmann & Härdle, 2026, Expert Systems with Applications)
  - Paper: "In the beginning was the Word: LLM-VaR and LLM-ES"; LLM forecasts benchmarked against classical risk models with formal backtests
  - Text models move from sentiment scores to the risk numbers themselves (Lecture 12)
- **BitMood: artificial intelligence (AI) analysis of Bitcoin trends via Facebook emotions** (Conda, Găman, Bâg, Mazurencu-Marinescu-Pele, Pele & Härdle, 2026, Digital Finance)
  - 120,000 Facebook posts, 2015–2023, classified with FinBERT into compound sentiment scores with the reactions of readers
  - Sentiment predicts Bitcoin trading volume; sentiment-based strategies outperform buy-and-hold; topic modelling tracks the themes of the discussion
- Findings as stated in the papers; numbers on the other slides come from our own computations.

(Bucharest University of Economic Studies; Romanian Academy, Institute for Economic Forecasting)

Explore it visually

- **Embedding Atlas (Apple)**: open-source tool that maps text embeddings in two dimensions, with automatic cluster labels, search and nearest neighbours
  - Run it on the PhraseBank sentences of this lecture and colour the points by sentiment.
- **Transformer Explainer**: the first step of a language model on your own sentence: tokens, token embeddings and the positional embeddings added to them.
- **3Blue1Brown, chapter 5**: directions in the embedding space, such as the difference between the vectors of “man and “woman.

Further reading and resources

- **Mikolov et al. (2013)** and **Pennington et al. (2014)**: the two embedding papers, both short.
- **Levy & Goldberg (2014)**: why word2vec is a matrix factorisation.
- **Loughran & McDonald (2011)**: the finance sentiment dictionary and why general dictionaries fail.
- **This course**: code, data, notebooks — github.com/danpele/NEURAL_BIZ.
- **Experiment yourself**: the notebook reproduces the charts of this lecture — notebooks/.
- **Seminar 6**: a sentiment index of ECB speeches built with your own classifier.

References (1/2)

- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*, ch. 12. Springer. doi:10.1007/978-3-031-45468-4
- Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T. (2017). Enriching word vectors with subword information. *TACL*, 5, 135–146. doi:10.1162/tacl_a_00051
- Conda, A. I., Găman, Ș., Båg, R. C., Mazurencu-Marinescu-Pele, M., Pele, D. T. & Härdle, W. K. (2026). BitMood: AI analysis of Bitcoin trends via Facebook emotions. *Digital Finance*, 8(3), 42. doi:10.1007/s42521-026-00205-3
- Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. In *Studies in Linguistic Analysis*, 1–32. Blackwell.
- Joulin, A., Grave, E., Bojanowski, P. & Mikolov, T. (2017). Bag of tricks for efficient text classification. *EACL*, 427–431. doi:10.18653/v1/E17-2068
- Levy, O. & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. *NIPS*, 27, 2177–2185. papers.nips.cc
- Loughran, T. & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *Journal of Finance*, 66(1), 35–65. doi:10.1111/j.1540-6261.2010.01625.x

References (2/2)

- Malo, P., Sinha, A., Korhonen, P., Wallenius, J. & Takala, P. (2014). Good debt or bad debt: Detecting semantic orientations in economic texts. *JASIST*, 65(4), 782–796. doi:10.1002/asi.23062
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *NIPS*, 26, 3111–3119. arXiv:1310.4546
- Pele, D. T., Bolovăneanu, V., Lin, M.-B., Ren, R., Ginavar, A. T., Spilak, B., Andrei, A.-V., Toma, F.-M., Lessmann, S. & Härdle, W. K. (2026). In the beginning was the Word: LLM-VaR and LLM-ES. *Expert Systems with Applications*, 295, 128676. doi:10.1016/j.eswa.2025.128676
- Pennington, J., Socher, R. & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP*, 1532–1543. doi:10.3115/v1/D14-1162
- Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513–523. doi:10.1016/0306-4573(88)90021-0
- Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

Appendix: the gradient of negative sampling

- For one pair, $\ell = -\log \sigma(s_o) - \sum_k \log \sigma(-s_k)$ with $s_o = \mathbf{u}_o^\top \mathbf{v}_w$, $s_k = \mathbf{u}_{n_k}^\top \mathbf{v}_w$.
- Using $\frac{d}{ds} \log \sigma(s) = 1 - \sigma(s)$: $\partial \ell / \partial \mathbf{v}_w = -(1 - \sigma(s_o))\mathbf{u}_o + \sum_k \sigma(s_k)\mathbf{u}_{n_k}$.
- $\partial \ell / \partial \mathbf{u}_o = -(1 - \sigma(s_o))\mathbf{v}_w$ and $\partial \ell / \partial \mathbf{u}_{n_k} = \sigma(s_k)\mathbf{v}_w$: real contexts are pulled closer, noise words pushed away.
- At the optimum, $\mathbf{u}_o^\top \mathbf{v}_w = \mathrm{PMI}(w, o) - \ln K$, with $\mathrm{PMI}(w, o) = \ln \frac{P(w,o)}{P(w)P(o)}$ (Levy & Goldberg, 2014).

Appendix: the models in code

```
tf = TfidfVectorizer(ngram_range=(1, 2), min_df=2, sublinear_tf=True).fit(train_text)
clf = LogisticRegression(C=C, class_weight='balanced', max_iter=3000).fit(tf.transform(train_text), y)

class EmbBag(nn.Module):                          # fastText-style classifier
    def __init__(self, n_vocab, dim, n_cls):
        super().__init__()
        self.emb = nn.EmbeddingBag(n_vocab, dim, mode='mean')
        self.out = nn.Linear(dim, n_cls)
    def forward(self, ids, offsets):
        return self.out(F.dropout(self.emb(ids, offsets), 0.3, self.training))
# SGNS step:  loss = -(logsigmoid((v*u_pos).sum(1)) + logsigmoid(-(U_neg @
v[...,None]).squeeze()).sum(1)).mean()
```

- Full code: lectures/L6/analysis_L6.py and the notebook of this lecture.