# Machine Learning for Natural Language Processing

*Guillaume Gravier*

guillaume.gravier@irisa.fr

ENSAI École nationale de la statistique et de l'analyse de l'information

# Outline of the course

**Lectures** (6 x 3h)

- Lecture #1: Introduction and representation of words
  - Notions: morphology, tokens, lemmas, POS, word net, word embedding
  - Hands-on: manipulate basic pipelines and visualize word embeddings
- Lecture #2: Representation of documents
  - Notions: vocabulary, Zipf's curse, bag of words, Bayes, RNN, BERT
  - Hands-on: basic tf-idf k-nn classifier
- Lecture #3: Language models
  - Notions: ngrams, LSTM, bi-LSTM, language generation
  - Hands-on: train a small LM and generate text
- Lecture #4: Transformers and large language models
  - Notions: encoder/decoder, transformers, fine-tuning
  - Hands-on: visualize embeddings, fine-tune a LLM

# Representation and classification of documents

# Representing documents: what for?

Documents can be (almost) everything ... that contains text

- book, chapter, paragraph, etc.
- newspaper/web article
- tweet, blog or facebook post

Most document representations seek to representing a document as a fixed-dimension *feature vector* further used for, e.g.,

- topic classification
- polarity and sentiment detection
- comparison of documents (information retrieval)

- often based on *the bag hypothesis*
  = order of words does not matter
- might implement selection of relevant terms

ENSAI École nationale de la statistique et de l'analyse de l'information

# A naive Bayes approach to document classification

Simplify the maximum a posteriori rule $p(c|d) = p(d|c)p(c)$ considering each $w \in d = \{w_1, \ldots, w_{n_d}\}$ independently, i.e.,

$$p(d|c) = \prod_{i=1}^{n_d} p(w_i|c)$$

T. Bayes (c. 1702–1761)

Estimating conditional word occurrence probabilities $p(w|c)$ from large corpora $D = \cup D_c$, e.g.,

$$p(w|c) = \frac{\sum_{d \in D_c} \delta(w,d)}{\sum_{v \in V} \sum_{d \in D_c} \delta(v,d)} \quad \text{or} \quad p(w|c) = \frac{\sum_{d \in D_c} n(w,d)}{\sum_{d \in D_c} n_d}$$

# The naive Bayes approach illustrated

1. class = love, content = {aimer: 5, manger: 0, Paul: 1, Virignie: 1, je: 5}
2. class = love, content = {aimer: 3, manger: 0, Paul: 0, Virignie: 0, je: 4}
3. class = food, content = {aimer: 0, manger: 2, Paul: 0, Virignie: 1, je: 5}
4. class = food, content = {aimer: 2, manger: 2, Paul: 0, Virignie: 0, je: 3}

For class 'love', we have:

$$P[\text{aimer}] = \frac{\sum_{d \in D_c} \delta(\text{aimer}, d)}{\sum_{v \in V} \sum_{d \in D_c} \delta(v, d)} = \frac{2}{6} \quad \text{or} \quad P[\text{aimer}] = \frac{\sum_{d \in D_c} n(\text{aimer}, d)}{\sum_{d \in D_c} n_d} = \frac{8}{19}$$

and for class 'food'

$$P[\text{aimer}] = \frac{\sum_{d \in D_c} \delta(\text{aimer}, d)}{\sum_{v \in V} \sum_{d \in D_c} \delta(v, d)} = \frac{1}{6} \quad \text{or} \quad P[\text{aimer}] = \frac{\sum_{d \in D_c} n(\text{aimer}, d)}{\sum_{d \in D_c} n_d} = \frac{2}{15}$$

# The naive Bayes approach illustrated

1. class = love, content = {aimer: 5, manger: 0, Paul: 1, Virignie: 1, je: 5}
2. class = love, content = {aimer: 3, manger: 0, Paul: 0, Virignie: 0, je: 4}
3. class = food, content = {aimer: 0, manger: 2, Paul: 0, Virignie: 1, je: 5}
4. class = food, content = {aimer: 2, manger: 2, Paul: 0, Virignie: 0, je: 3}

With the first estimator, we get in the end

| class | P[aimer] | P[manger] | P[Paul] | P[Virginie] | P[je] |
|---|---|---|---|---|---|
| love | 2/6 | 0/6 | 1/6 | 1/6 | 2/6 |
| food | 1/6 | 2/6 | 0/6 | 1/6 | 2/6 |

Assuming equal class prior, classify new document $d$ = {aimer: 2, manger: 0, Paul: 0, Virignie: 1, je: 1} according to

$$\begin{aligned} P[d|\text{class=love}] &= 0.5 * (2*2/6) * 1/6 * 2/6 \sim .0185 \\ P[d|\text{class=food}] &= 0.5 * (2*1/6) * 1/6 * 2/6 \sim .0093 \end{aligned}$$

# Naive Bayes and regularization (aka smoothing)

Now classifying $d$ = {aimer: 0, manger: 10, Paul: 1, Virignie: 0, je: 0} :

$$\begin{aligned} P[d|\text{class=love}] &= 0.5 * (10 * 0) * 1/6 = 0 \\ P[d|\text{class=food}] &= 0.5 * (10 * 2/6) * 0 = 0 \end{aligned}$$

Need for smoothed probability estimates to avoid 0s, e.g,

$$p(w|c) = \frac{1 + \sum_{d \in D_c} n(w,d)}{|V| + \sum_{d \in D_c} n_d} \quad \text{or} \quad p(w|c) = \frac{\lambda P[w] + \sum_{d \in D_c} n(w,d)}{\lambda + \sum_{d \in D_c} n_d}$$

with $P[W] \rightsquigarrow \text{Dir}(\alpha)$.

# Why smoothing is so important? (because of Zipf)

Statistics on the newspaper Le Monde in 2003

| $r$ | $n_r$ | token | $r$ | $n_r$ | token |
|---|---|---|---|---|---|
| 1 | 227306 | académisé | 274928 | 1 | pour |
| 2 | 59053 | gutturale | 286277 | 1 | une |
| 3 | 28459 | port-cros | 287036 | 1 | dans |
| 4 | 17223 | s'imputer | 325378 | 1 | a |
| 5 | 11483 | remariée | 339432 | 1 | un |
| 6 | 8310 | échangée | 438658 | 1 | du |
| 7 | 6190 | mastercard | 494437 | 1 | en |
| 8 | 4901 | délégitimer | 591394 | 1 | des |
| 9 | 3744 | teenage | 638864 | 1 | et |
| 10 | 3072 | diamonds | 682522 | 1 | à |
| 11 | 2477 | matta | 684617 | 1 | les |
| 12 | 2022 | cammas | 836026 | 1 | le |
| 13 | 16462 | collabos | 1081822 | 1 | la |
| 14 | 7458 | sidibe | 1892396 | 1 | de |

[courtesy of François Yvon]

George K. Zipf

1902–1950

Frequent events are rare and rare events are frequent, which roughly translate to

$$\text{rank}(w)\text{freq}(w) = \text{cst}$$

ENSAI École nationale de la statistique et de l'analyse de l'information

# Explicit bag-of-words: the vector space model

Assign a weight to each possible term (token) in a fixed-size vocabulary according to its appearance in the document

The bag of words principle is a powerful principle to represent documents.

$\xrightarrow[\textit{normalize}]{\textit{select}}$

bag
word
principle
powerful
represent
document

$\xrightarrow[\textit{weight}]{\textit{count}}$

| | |
|---|---|
| 0 | *abalone* |
| … | …. |
| 1 | *bag* |
| 1 | *document* |
| … | … |
| 1 | *word* |
| … | … |
| 1 | *powerful* |
| 2 | *principle* |
| … | … |
| 1 | *represent* |
| … | … |
| 0 | *zygote* |

# Choosing and weighting representation terms

## Step 1. Selection of terms for the vocabulary

- tokenization and normalization
- lemmatization, stemming ... or none
- selection of relevant terms
  - frequency, POS (NVA), stop lists
  - might be crucial (retrieval) ... or not (classification)

## Step 2. Assignement of weights for each token

- binary indicator $\delta(w, d)$ $\longrightarrow$ aka 1-hot encoding
- number of occurrences $n(w, d)$ of word $w$ in document $d$
- frequency of occurrence $n(w, d) / \sum_{v \in V} n(v, d)$

  $\Rightarrow$ issue with frequent words, typically non-informative function words

ENSAI École nationale de la statistique et de l'analyse de l'information

# The tf-idf weighting scheme

Normalizing term frequency to downplay frequent function words that bear limited information in most cases

$$f(w,d) = \underbrace{\left( \frac{n(w,d)}{\sum_{v \in V} n(v,d)} \right)}_{\text{term frequency}} \underbrace{\log \left( \frac{\sum_{d' \in D} \delta(w,d')}{N} \right)^{-1}}_{\text{inverse document frequency}}$$

where $D$ is a collection of $N$ documents to compute prior probability of how likely $w$ is to appear in a document

$\Rightarrow$ can be extended in a number of ways mixing local weight (term frequency), global weight (inverse document frequency) and possibly a normalization weight (to account for different document length for instance)

ENSAI École nationale de la statistique et de l'analyse de l'information

## tf-idf illustrated

1. class = love, content = {aimer: 5, manger: 0, Paul: 1, Virignie: 1, je: 5}
2. class = love, content = {aimer: 3, manger: 0, Paul: 0, Virignie: 0, je: 4}
3. class = food, content = {aimer: 0, manger: 2, Paul: 0, Virignie: 1, je: 5}
4. class = food, content = {aimer: 2, manger: 2, Paul: 0, Virignie: 0, je: 3}

- aimer appears in 3 documents out of |D|=4

$$\text{idf}(\text{aimer}) = \log\left(\frac{\sum_{d' \in D} \delta(\text{aimer}, d')}{|D|}\right)^{-1} = \log(4/3) \simeq 0.125$$

- aimer appears 5 times in document $d_1$

$$\text{tf}(\text{aimer}, d_1) = \frac{n(\text{aimer}, d_1)}{\sum_{v \in V} n(v, d_1)} = 5/12 \simeq 0.417$$

- weight of aimer in document $d_1 = \text{tf}(\text{aimer}, d_1)\text{idf}(\text{aimer}) \simeq 0.052$

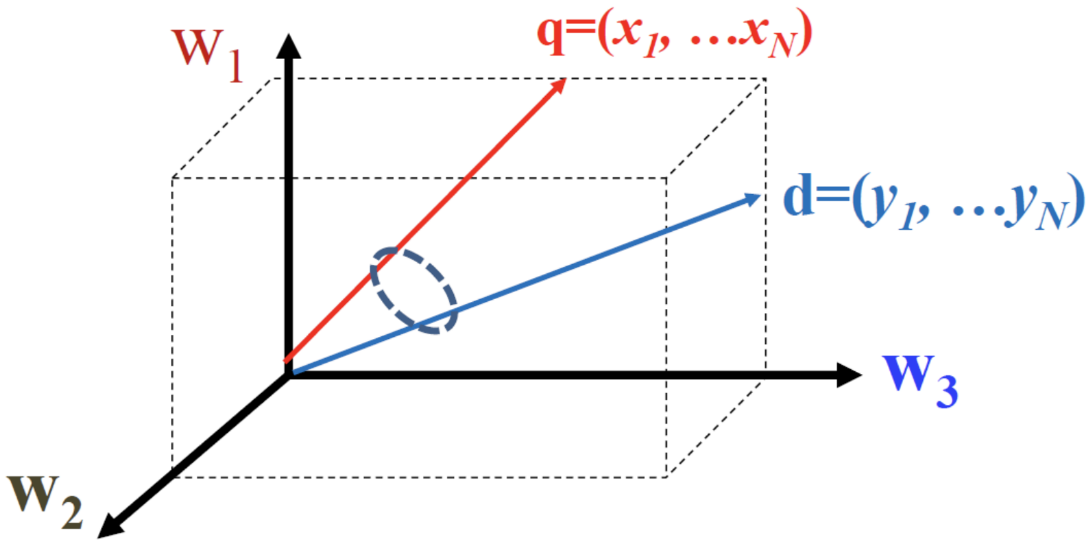
# tf-idf illustrated

1. class = love, content = {aimer: 5, manger: 0, Paul: 1, Virignie: 1, je: 5}
2. class = love, content = {aimer: 3, manger: 0, Paul: 0, Virignie: 0, je: 4}
3. class = food, content = {aimer: 0, manger: 2, Paul: 0, Virignie: 1, je: 5}
4. class = food, content = {aimer: 2, manger: 2, Paul: 0, Virignie: 0, je: 3}

| | idf | doc1 | doc2 | doc3 | doc4 |
|---|---|---|---|---|---|
| $n_w$ | | 12 | 7 | 8 | 7 |
| aimer | 0.125 | 0.052 | 0.054 | 0 | 0.036 |
| manger | 0.301 | 0 | 0 | 0.077 | 0.089 |
| Paul | 0.602 | 0.050 | 0 | 0 | 0 |
| Virginie | 0.301 | 0.025 | 0 | 0.038 | 0 |
| je | 0 | 0 | 0 | 0 | 0 |

# The vector space model (information retrieval)

Documents (and possibly queries in IR) are represented in a vector space over which we can define a metric




borrowed from Tonny Kwon's blog

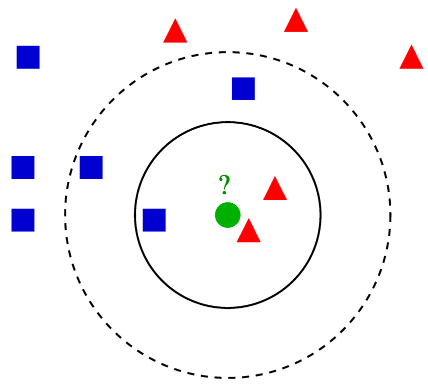
dot product $$x \cdot y = \sum_i x_i y_i$$

$\ell^2$ norm $$||x - y|| = \sqrt{\sum_i (x_i - y_i)^2}$$

cosine $$\text{cosine}(x, y) = \frac{x \cdot y}{||x||\ ||y||}$$

# Classification in the vector space model

All flavors of feature-based classifiers can be used with the bag-of-word representation, e.g.,




© Antti Ajanki

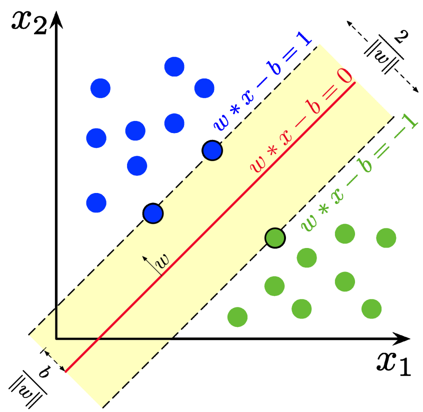
- k-nearest neighbors
- logistic regression

$$p(c|d) = \frac{1}{1 + \exp\left(\alpha_0 + \sum_{w \in d} \alpha_w f(w, d)\right)}$$

- support vector machines

$$\widehat{c} = \text{sign}\left(\sum_{w \in V} \alpha_w f(w, d) - \alpha_0\right)$$

- feed-forward neural nets




© Larhmam

# Latent variable variants of the BoW model

Some of the downsides of the BoW approach

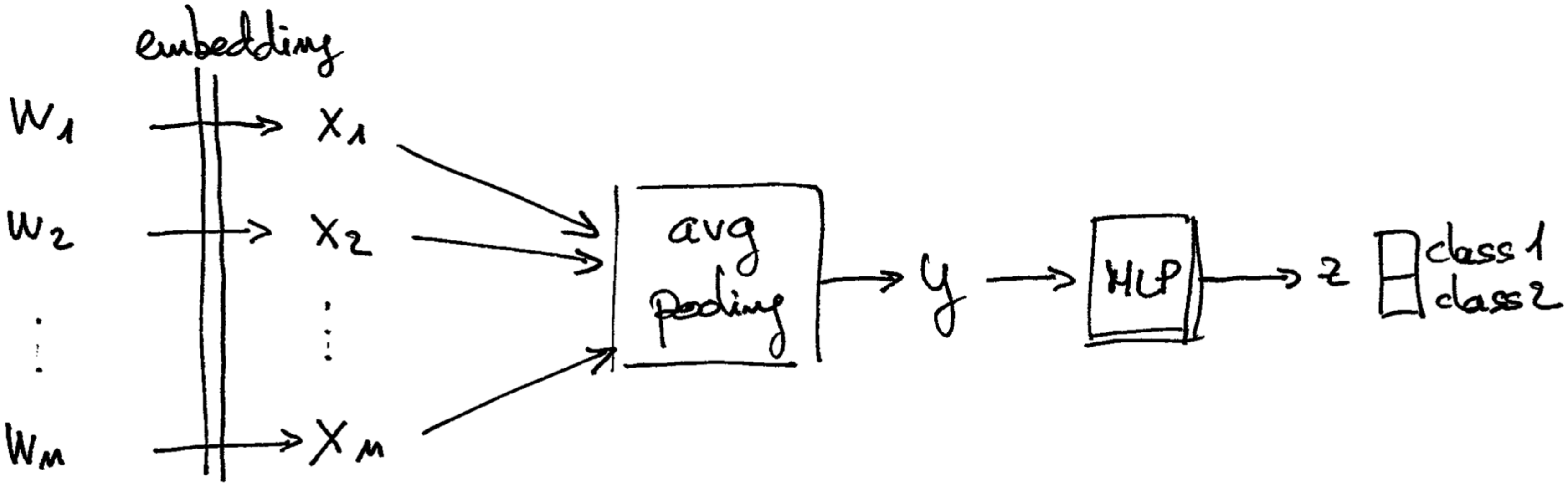
- no ordering of words     that's the price to pay
- very sparse representation, high dimension
- distributional semantics is absent (cat $\neq$ kitty)
- cannot compare documents with no words in common

Seek small, compact and efficient representations that can be directly used rather than the BoW vector

Option 1: Latent semantic indexing with PCA/SVD

Option 2: Latent Dirichlet allocation

# A naive average word embedding approach




$x_i = W \cdot \mathbb{1}hot(w_i) \in \mathbb{R}^d$  $y = \frac{1}{n} \sum_i x_i$  $z = \mathrm{softmax}(\tanh(Ay + b)) \in \mathbb{R}^2$

Is it distributional semantics if we train the whole thing?

# A naive average word embedding approach implemeted

See notebook for details.

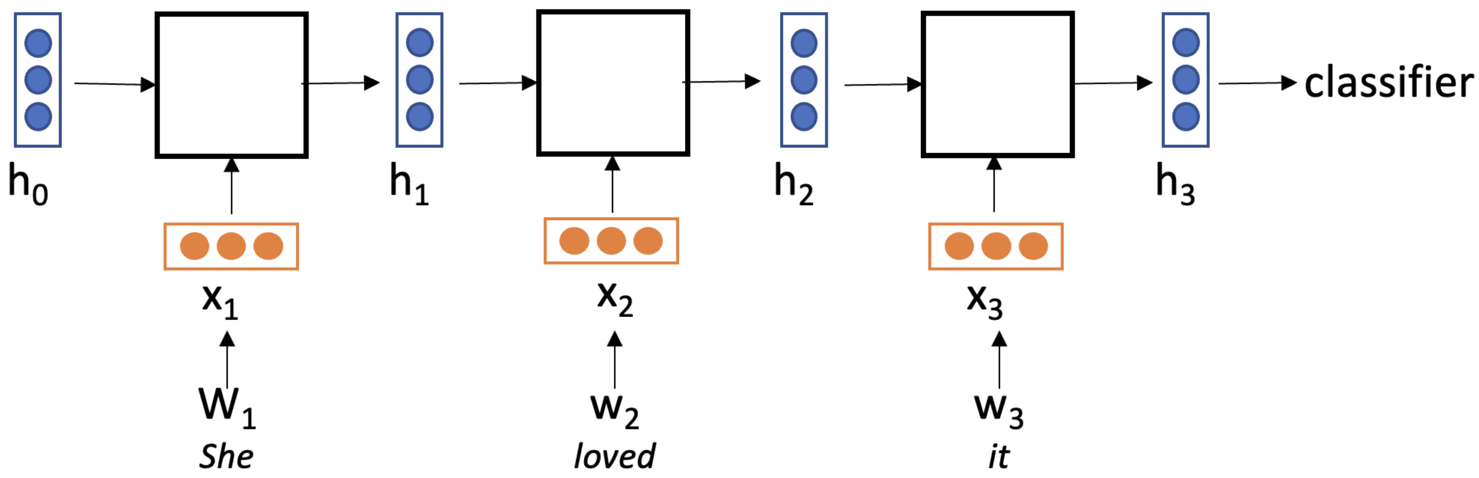
```
class NLPAvgPooling(torch.nn.Module):

[...]

  def forward(self, **kwargs):
    x = self.embedding(kwargs['ids']) # batch_size * maxlen * dim
    x = torch.mean(x, dim=1)          # batch_size * dim
    x = self.softmax(self.linear(x))  # batch_size * nclasses
```

# Embedding sequences with recurrent neural networks

$h_i$ **= summary of document up to** $w_i \Rightarrow h_n$ **= summary of document**




Example of an Elman recurrent network

Embedding layer $x_i = c(w_i)$

Merging layer $y_i = x_i + h_{i-1}$

State prediction $h_i = \sigma(U y_i)$ or $\sigma(U_c c_i + U_h h_{i-1}))$

# RNN utterance embedding implementation

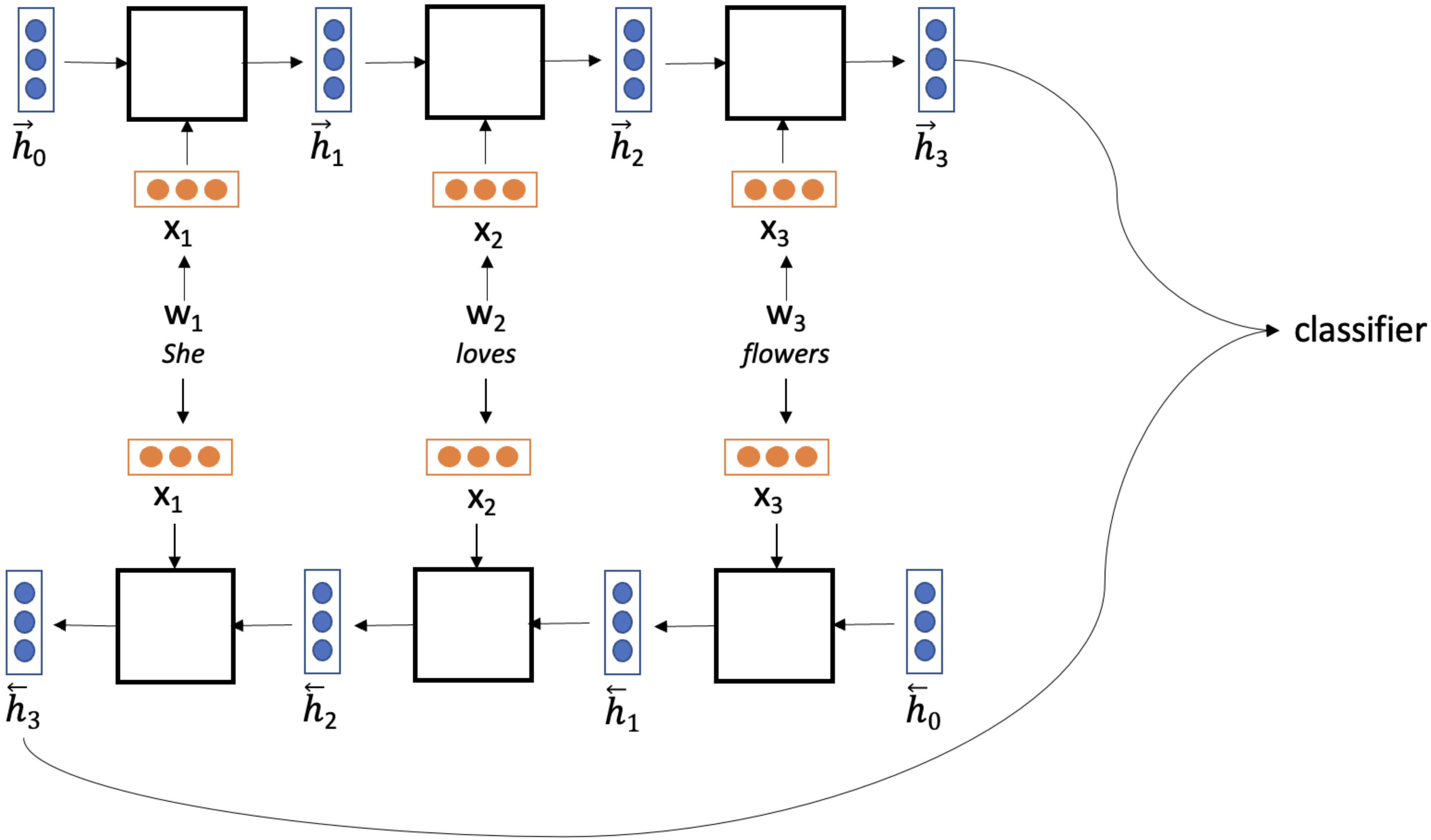
See notebook for details.

```
class NLPRNN(torch.nn.Module):

[...]

  def forward(self, **kwargs):
    x = self.embedding(kwargs['ids'])   # batch_size * maxlen * dim
    _, (x, _) =  self.lstm(x)           # 1 * batch_size * dim
    x = self.softmax(self.linear(x[0])) # batch_size * nclasses
```

# Bidirectionnal sequence embedding



# Sentence embedding with RNNs: evaluation

Intrinsic evaluations, e.g., Semantic Textual Similarity Benchmark

| | | |
|---|---|---|
| Other ways are needed | We must find other ways | 4.4 |
| I absolutely do believe there was an iceberg in those waters | I don't believe there was any iceberg at all anywhere near the Titanic | 1.2 |

Extrinsic / task-based evaluation, e.g., GLUE for English ...

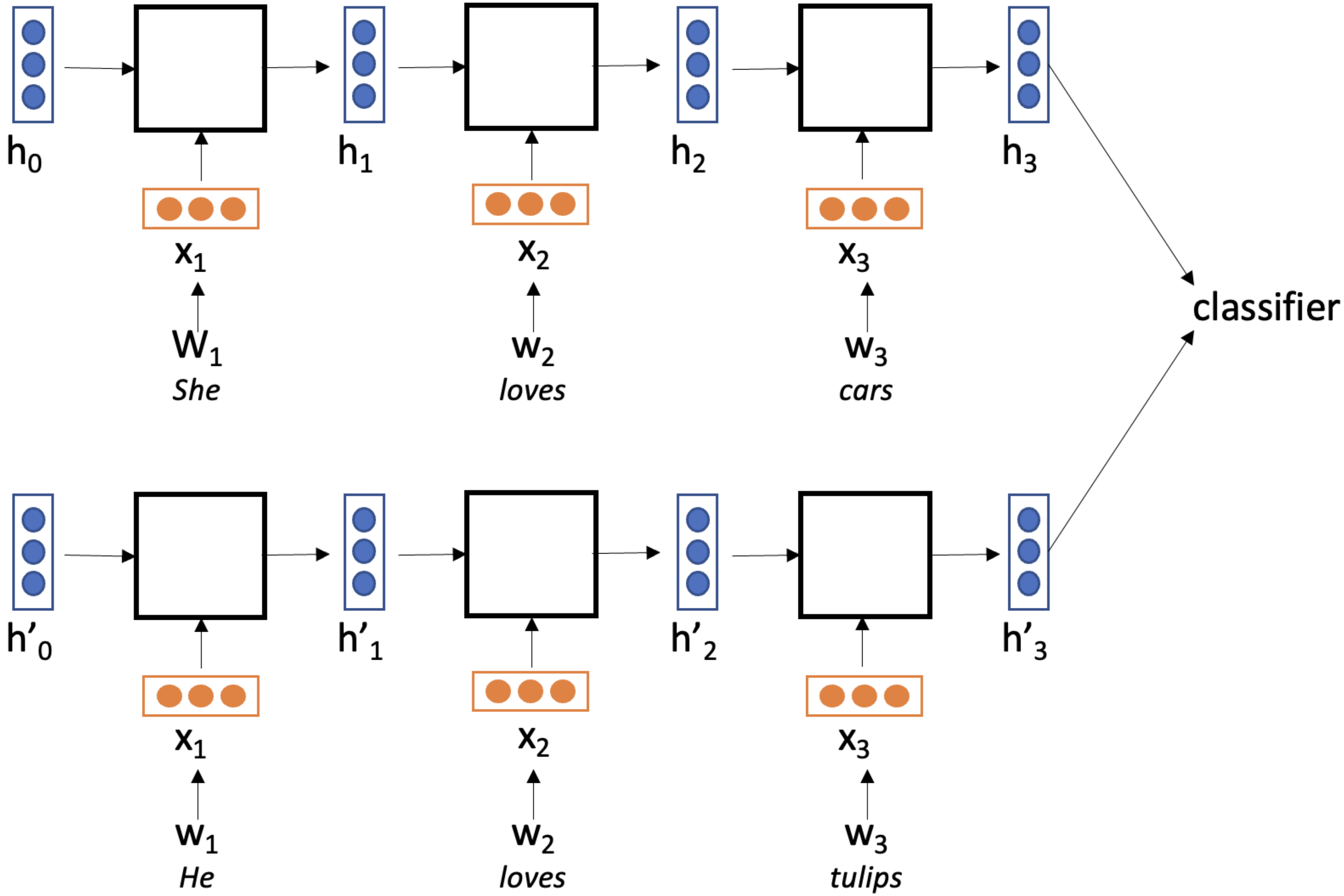
| | |
|---|---|
| Corpus of Linguistic Acceptability | sentence is grammatical or not |
| Stanford Sentiment Treebank | valence prediction |
| Microsoft Research Paraphrase | semantically equivalent or not |
| Quora Question Pairs | semantically equivalent or not |
| Multi-Genre Natural Language Inference | predict entailment |
| Recognizing Textual Entailment | |
| Stanford Question Answering | paragraph contains answer or not |
| Winograd Schema Challenge | reference prediction (closed list) |

Alex Wang et al., 2018. GLUE: A multi-task benchmark and analysis platform for NLU

... or FLUE, the recent French equivalent of GLUE

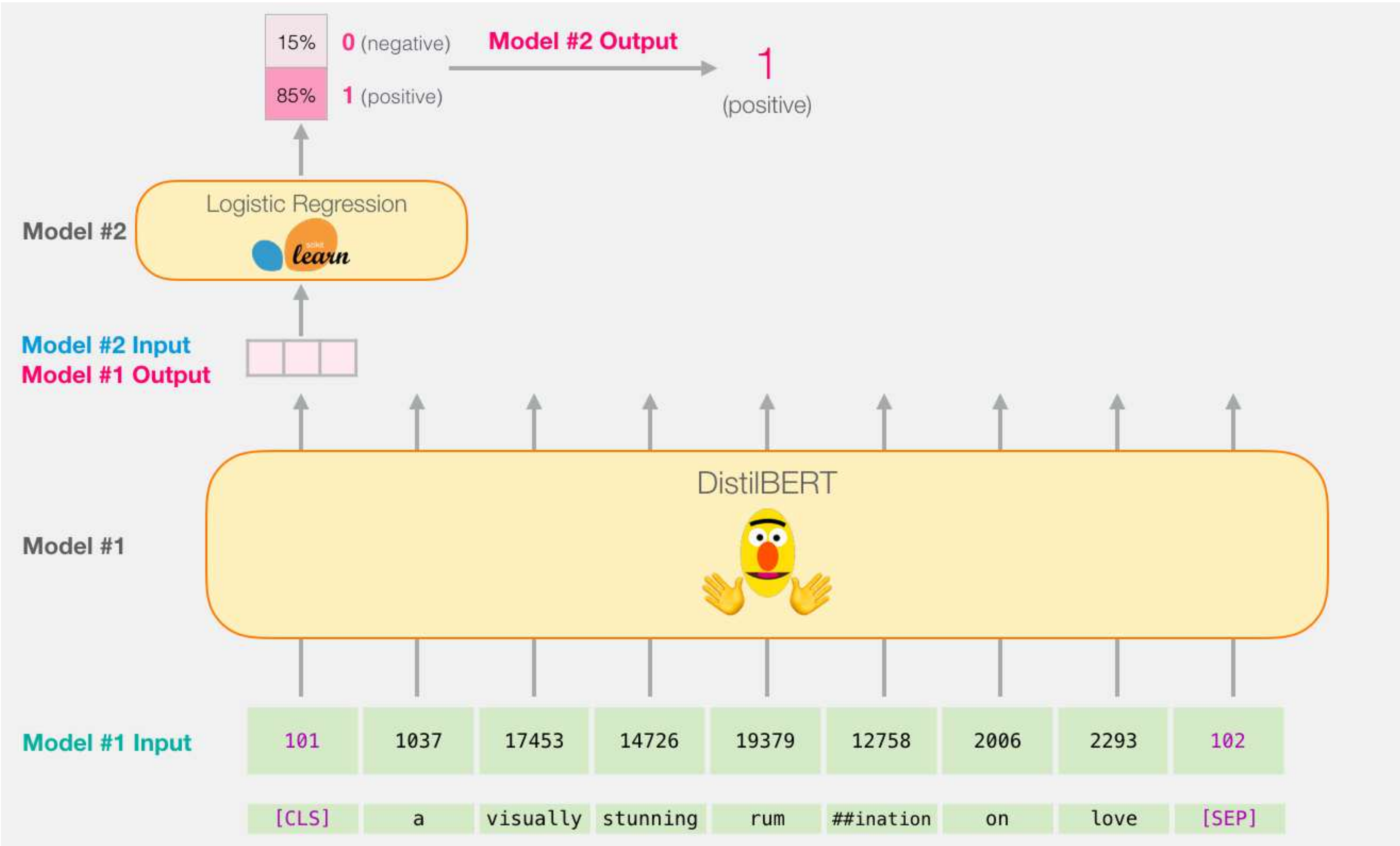
Hang Le et al., 2019. FlauBERT: Unsupervised language model pre-training for French

# Shades of RNNs for sequence embedding: comparing two inputs



# What with BERT-like models?

Use BERT pre-trained model to embed document and use the embeding through a classifier (one can train the whole thing at once).




That's all Folks!