Session 3
Word-embeddings approaches and large language models

Michele Scotto di Vettimo
King's College London

https://mscottodivettimo.github.io/
michele.scotto_di_vettimo@kcl.ac.uk

LISS2117 · *Quantitative methods for text classification and topic detection*

Programme

- Large language models in short
- Embeddings representations
- Classification with word-embeddings
- Transformers and classification with LLMs
- Multilingual text classification

Recap of previous session

In previous episodes...

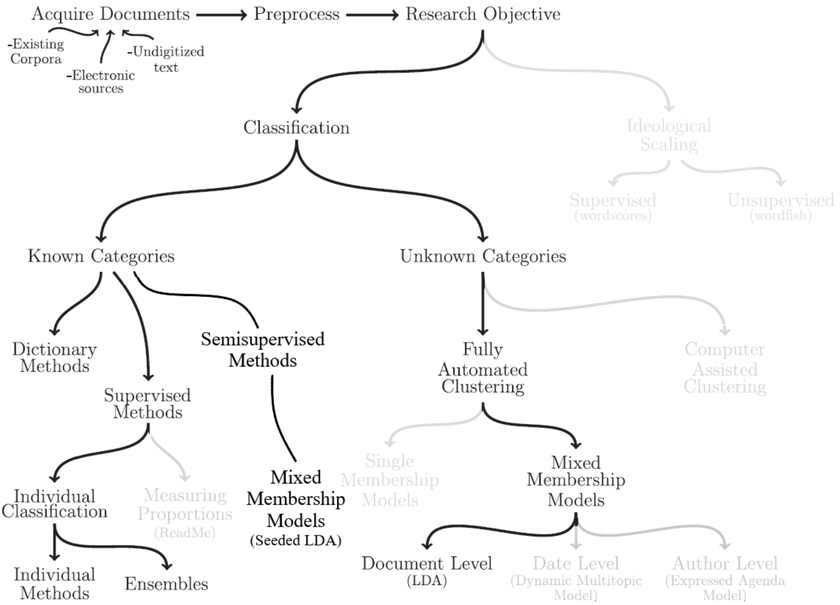


Fig. 1 An overview of text as data methods.

In previous episodes...

In the last session we covered different types of unsupervised, semisupervised and supervised methods for classification

- Unsupervised: LDA and STM
- Semisupervised: Keyword-assisted topic models
- Supervised: Machine learning algorithms

In previous episodes...

In the last session we covered different types of unsupervised, semisupervised and supervised methods for classification

- Unsupervised: LDA and STM
- Semisupervised: Keyword-assisted topic models
- Supervised: Machine learning algorithms

All such methods rely predominantly on bag-of-words representation of texts

However, they differ in terms of their overall aim and the amount of input data that needs to be provided by the analyst to make them work

In previous episodes...

In the last session we covered different types of unsupervised, semisupervised and supervised methods for classification

- Unsupervised: LDA and STM
- Semisupervised: Keyword-assisted topic models
- Supervised: Machine learning algorithms

All such methods rely predominantly on bag-of-words representation of texts

However, they differ in terms of their overall aim and the amount of input data that needs to be provided by the analyst to make them work

Best method depends on task (and data), but all techniques need to be validated somehow

Various performance measures are available, some of them useful in summarising how good our model is at replicating human coding or some benchmark data

Large language models (LLMs): Intro

Large language models (LLMs): Intro

LLMs are artificial intelligence systems trained on massive amounts of textual data

They learn to predict and generate human-like language by identifying patterns, relations, and structures in natural language

Large language models (LLMs): Intro

LLMs are artificial intelligence systems trained on massive amounts of textual data

They learn to predict and generate human-like language by identifying patterns, relations, and structures in natural language

- LLMs are based on neural networks and so called "transformer" architecture (more on this below)
- Words and sentences are represented in numerical vectors (called embeddings)
- The model then learns statistical relations between words (technically, their embeddings) so as to produce a "context-aware" understanding of the meaning
- Then, the model uses the learned patterns to perform tasks like translation, summarization, or classification

Large language models (LLMs): Intro

LLMs are artificial intelligence systems trained on massive amounts of textual data

They learn to predict and generate human-like language by identifying patterns, relations, and structures in natural language

- LLMs are based on neural networks and so called "transformer" architecture (more on this below)
- Words and sentences are represented in numerical vectors (called embeddings)
- The model then learns statistical relations between words (technically, their embeddings) so as to produce a "context-aware" understanding of the meaning
- Then, the model uses the learned patterns to perform tasks like translation, summarization, or classification

In doing so, they can be used to perform tasks that, traditionally, fall under the domain of quantitative text analysis (e.g., sentiment analysis, topic labeling, topic detection, classification)

Popular examples of LLMs include OpenAI's GPT, Google's PaLM, and Meta's LLaMA

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

In the context of text classification, LLMs can:

- Directly classify texts using prompt-based methods (e.g., "Classify this tweet as positive or negative")

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

In the context of text classification, LLMs can:

- Directly classify texts using prompt-based methods (e.g., "Classify this tweet as positive or negative")
- Be fine-tuned on labeled datasets for more accuracy and task-specific performance

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

In the context of text classification, LLMs can:

- Directly classify texts using prompt-based methods (e.g., "Classify this tweet as positive or negative")
- Be fine-tuned on labeled datasets for more accuracy and task-specific performance

In short, they can do most of the tasks that, until a couple of years ago, would be implemented with bag-of-words based approaches

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

In the context of text classification, LLMs can:

- Directly classify texts using prompt-based methods (e.g., "Classify this tweet as positive or negative")
- Be fine-tuned on labeled datasets for more accuracy and task-specific performance

In short, they can do most of the tasks that, until a couple of years ago, would be implemented with bag-of-words based approaches

Pros:

- Minimal pre-processing required

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

In the context of text classification, LLMs can:

- Directly classify texts using prompt-based methods (e.g., "Classify this tweet as positive or negative")
- Be fine-tuned on labeled datasets for more accuracy and task-specific performance

In short, they can do most of the tasks that, until a couple of years ago, would be implemented with bag-of-words based approaches

Pros:

- Minimal pre-processing required
- Fewer examples are needed to learn a task

Large language models (LLMs): Intro

LLMs can be fine-tuned or prompted to perform quantitative text analysis

In the context of text classification, LLMs can:

- Directly classify texts using prompt-based methods (e.g., "Classify this tweet as positive or negative")
- Be fine-tuned on labeled datasets for more accuracy and task-specific performance

In short, they can do most of the tasks that, until a couple of years ago, would be implemented with bag-of-words based approaches

Pros:

- Minimal pre-processing required
- Fewer examples are needed to learn a task
- Same model can be instructed to perform multiple tasks

Transfer learning

Perhaps the most important advantage and strength of LLMs is that they enable "transfer learning"

Transfer learning

Perhaps the most important advantage and strength of LLMs is that they enable "transfer learning"

Transfer learning $\neq$ Machine learning

Transfer learning

Perhaps the most important advantage and strength of LLMs is that they enable "transfer learning"

Transfer learning $\neq$ Machine learning

Machine learning:

- We train a model from scratch (on our training data) to perform a similar task
- The model has no "language knowledge" (i.e., understanding of the semantic relationships between words)
- The model has no "task knowledge" either, we need to train it

Transfer learning

Perhaps the most important advantage and strength of LLMs is that they enable "transfer learning"

Transfer learning $\neq$ Machine learning

Machine learning:

- We train a model from scratch (on our training data) to perform a similar task
- The model has no "language knowledge" (i.e., understanding of the semantic relationships between words)
- The model has no "task knowledge" either, we need to train it

Transfer learning:

- We use a pre-trained model, and adapt ("fine-tune") it to our training data, to perform a specific task
- By being pre-trained on a large amount of texts, LLMs acquire an understanding of sematic relationship between words ("language knowledge")

Transfer learning

Perhaps the most important advantage and strength of LLMs is that they enable "transfer learning"

Transfer learning $\neq$ Machine learning

Machine learning:

- We train a model from scratch (on our training data) to perform a similar task
- The model has no "language knowledge" (i.e., understanding of the semantic relationships between words)
- The model has no "task knowledge" either, we need to train it

Transfer learning:

- We use a pre-trained model, and adapt ("fine-tune") it to our training data, to perform a specific task
- By being pre-trained on a large amount of texts, LLMs acquire an understanding of sematic relationship between words ("language knowledge")

In transfer learning we build on the model's pre-existing language knowledge

With fine-tuning on our data, we allow the model to acquire task-specific knowledge to better perform our task

Embeddings representations

Embeddings representations

But where does this language knowledge of LLMs come from?

Embeddings representations

But where does this language knowledge of LLMs come from?

Embeddings representation provide "language knowledge"

Embeddings representations

But where does this language knowledge of LLMs come from?

Embeddings representation provide "language knowledge"

Contrary to a bag-of-words representation, an embeddings representation captures semantic relations among words

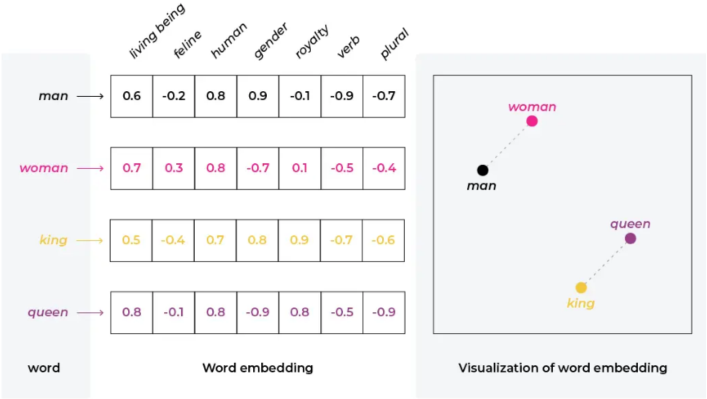
Embeddings representations

But where does this language knowledge of LLMs come from?

Embeddings representation provide "language knowledge"

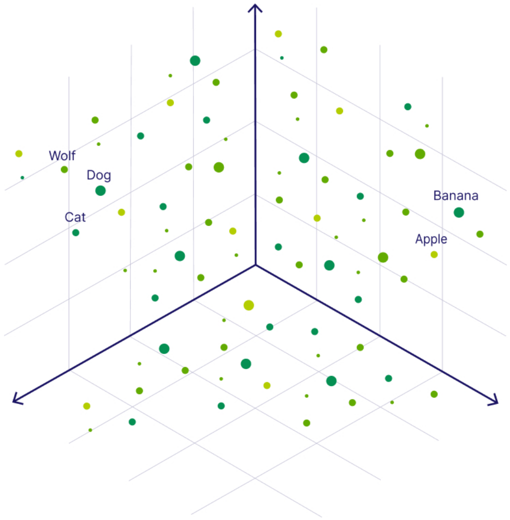
Contrary to a bag-of-words representation, an embeddings representation captures semantic relations among words

Words are represented by vectors of numbers, describing their position in the semantic space



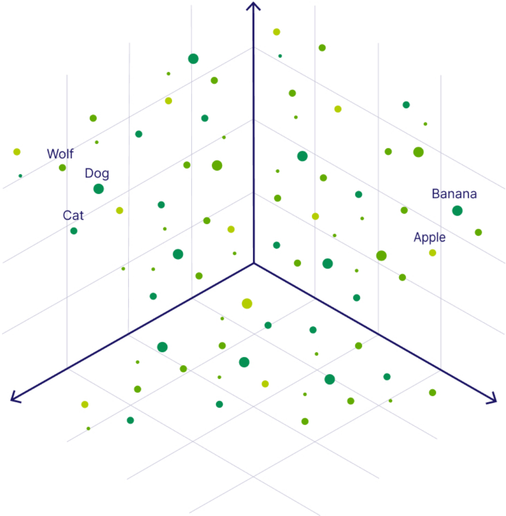
Embeddings representations

Words with similar meanings have similar vectors, hence they are "close" to each other in a multi-dimensional semantic space



Embeddings representations

Words with similar meanings have similar vectors, hence they are "close" to each other in a multi-dimensional semantic space



LLMs construct these vectors and map the semantic space by being trained on millions of texts

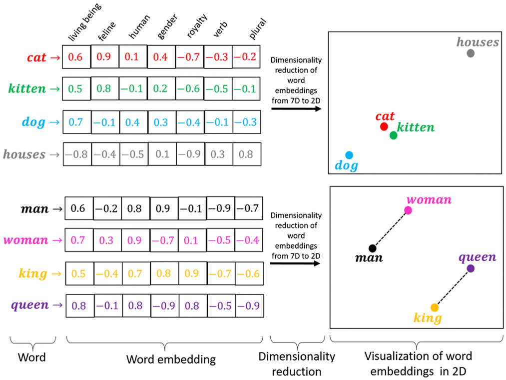
Embeddings representations

Each word vector will have many dimensions. You can think of a dimension as a feature of the word: a learned value that captures some aspects of usage

Embeddings representations

Each word vector will have many dimensions. You can think of a dimension as a feature of the word: a learned value that captures some aspects of usage

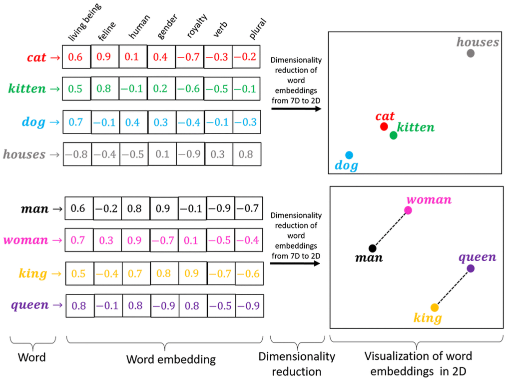
By converting words to numerical vectors, we embed semantic information into such vectors (hence the term "word-embeddings")



Embeddings representations

Each word vector will have many dimensions. You can think of a dimension as a feature of the word: a learned value that captures some aspects of usage

By converting words to numerical vectors, we embed semantic information into such vectors (hence the term "word-embeddings")



So a 768-dimensional embedding means that each word is represented by a 768-number vector, where each number encodes part of the word's linguistic or semantic context (768 is the number of dimensions used by base BERT models)

Supervised classification with word-embeddings

Word-embeddings and machine learning algorithms

Embeddings representations are not just the input for more complex models (such as LLMs, see below). They can also be used as input for more traditional classification methods

Word-embeddings and machine learning algorithms

Embeddings representations are not just the input for more complex models (such as LLMs, see below). They can also be used as input for more traditional classification methods

Pre-computed word embeddings can be paired with conventional machine-learning classifiers:

Word-embeddings and machine learning algorithms

Embeddings representations are not just the input for more complex models (such as LLMs, see below). They can also be used as input for more traditional classification methods

Pre-computed word embeddings can be paired with conventional machine-learning classifiers:

- Choose the embeddings

Word-embeddings and machine learning algorithms

Embeddings representations are not just the input for more complex models (such as LLMs, see below). They can also be used as input for more traditional classification methods

Pre-computed word embeddings can be paired with conventional machine-learning classifiers:

- Choose the embeddings
- Turn each text into a single numerical vector

Word-embeddings and machine learning algorithms

Embeddings representations are not just the input for more complex models (such as LLMs, see below). They can also be used as input for more traditional classification methods

Pre-computed word embeddings can be paired with conventional machine-learning classifiers:

- Choose the embeddings
- Turn each text into a single numerical vector
- Feed the text vectors to a traditional classifier (e.g., a random forest)

Word-embeddings and machine learning algorithms

Embeddings representations are not just the input for more complex models (such as LLMs, see below). They can also be used as input for more traditional classification methods

Pre-computed word embeddings can be paired with conventional machine-learning classifiers:

- Choose the embeddings
- Turn each text into a single numerical vector
- Feed the text vectors to a traditional classifier (e.g., a random forest)
- Train and validate as with any other machine learning task

We will be mostly relying on the `word2vec` package:

- `word2vec()`

Then, we will use once again the `randomForest()` function to classify texts using word vectors

When lost, cry for `help()`!

Transformers

Embeddings and Transformers

Once our words are converted into vectors (embeddings), these embeddings serve as input to "transformers" models

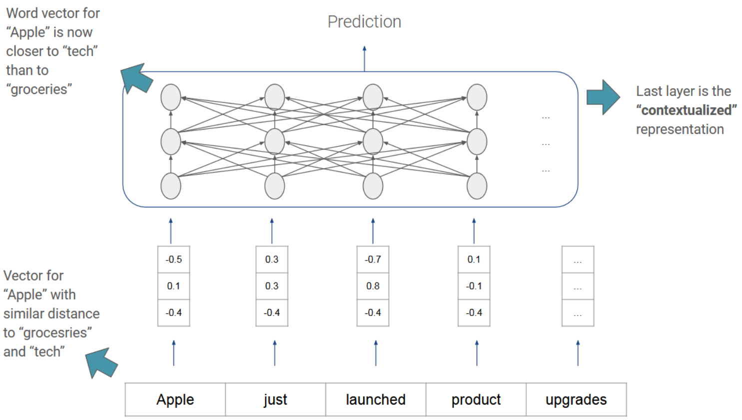
Transformers are a type of deep learning model architecture designed to process sequences of data (embeddings) to add contextual information

Embeddings and Transformers

Once our words are converted into vectors (embeddings), these embeddings serve as input to "transformers" models

Transformers are a type of deep learning model architecture designed to process sequences of data (embeddings) to add contextual information

The embeddings go through different "layers", where their vectors are updated and adapted into "contextual embeddings"



Embeddings and Transformers

This learning process is what allows LLMs to acquire "language knowledge": representing 'apple' differently in context of 'groceries' and 'technology'

The core elements of this learning process are masked language modelling, the self-attention mechanism, and positional encoding

Embeddings and Transformers

This learning process is what allows LLMs to acquire “language knowledge”: representing ‘apple’ differently in context of ‘groceries’ and ‘technology’

The core elements of this learning process are masked language modelling, the self-attention mechanism, and positional encoding

- Masked language modelling: some tokens in the input are replaced with “[MASK]”, and the model is trained to predict them based on context

Embeddings and Transformers

This learning process is what allows LLMs to acquire “language knowledge”: representing ‘apple’ differently in context of ‘groceries’ and ‘technology’

The core elements of this learning process are masked language modelling, the self-attention mechanism, and positional encoding

- Masked language modelling: some tokens in the input are replaced with “[MASK]”, and the model is trained to predict them based on context
- Self-attention mechanism: each word’s representation is updated based on all other words in the sentence. The model looks at unmasked tokens to infer what word fits best in the masked position

Embeddings and Transformers

This learning process is what allows LLMs to acquire "language knowledge": representing 'apple' differently in context of 'groceries' and 'technology'

The core elements of this learning process are masked language modelling, the self-attention mechanism, and positional encoding

- Masked language modelling: some tokens in the input are replaced with "[MASK]", and the model is trained to predict them based on context
- Self-attention mechanism: each word's representation is updated based on all other words in the sentence. The model looks at unmasked tokens to infer what word fits best in the masked position
- Positional encoding: adds information about token positions so that the model knows which token comes first, second, etc.

Context aware embeddings

Embeddings representations

Not only embeddings representations capture context-specific meaning ("seal" the animal $\neq$ "seal" for stamping), they also allow LLMs to make the most out of the learned "language knowledge"

Embeddings representations

Not only embeddings representations capture context-specific meaning ("seal" the animal $\neq$ "seal" for stamping), they also allow LLMs to make the most out of the learned "language knowledge"

Imagine we train a traditional bag-of-words based model on some texts. If the texts we want to classify have tokens that are not present in the training texts, the model does not know how to use those tokens for labelling purposes

Embeddings representations

Not only embeddings representations capture context-specific meaning ("seal" the animal $\neq$ "seal" for stamping), they also allow LLMs to make the most out of the learned "language knowledge"

Imagine we train a traditional bag-of-words based model on some texts. If the texts we want to classify have tokens that are not present in the training texts, the model does not know how to use those tokens for labelling purposes

On the contrary, by relying on their mapping of the semantic space, LLMs can use the information contained in the word-embeddings for classification purposes, even if the tokens are not in our training data

Embeddings representations

Example:

- Training texts:
  1. "I live in London" → "United Kingdom"
  2. "They work in Paris" → "France"
- Predict: "We are travelling to Manchester" → ?

Embeddings representations

Example:

- Training texts:
  1. "I live in London" → "United Kingdom"
  2. "They work in Paris" → "France"
- Predict: "We are travelling to Manchester" → ?

- Classic bag-of-words approach can't map any token in the new text to the labels

Embeddings representations

Example:

- Training texts:
  1. "I live in London" → "United Kingdom"
  2. "They work in Paris" → "France"
- Predict: "We are travelling to Manchester" → ?

- Classic bag-of-words approach can't map any token in the new text to the labels
- Word-embedding approach will figure out that the vector of "Manchester" is closer to the vector for "London" than to the vector for "Paris", and assing the label accordingly

Embeddings representations

Example:

- Training texts:
  1. "I live in London" → "United Kingdom"
  2. "They work in Paris" → "France"
- Predict: "We are travelling to Manchester" → ?

- Classic bag-of-words approach can't map any token in the new text to the labels
- Word-embedding approach will figure out that the vector of "Manchester" is closer to the vector for "London" than to the vector for "Paris", and assing the label accordingly

Given that such models possess some prior knowledge of the language, they can be used in universal tasks (i.e., a task format that does not require task-specific adaptation) without further fine-tuning (i.e., without providing them with additional task knowledge)

We will run our LLMs in Python language via Google Colab

- 1. Getting started
- 2. Transformers
- 3. Universal tasks

We will use Colab to run models stored on the Hugging Face repository. For this, you need to register a free account and create an access token (see how to create a token here).

Classification with transformers models

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the “task knowledge” that complements its pre-existing “language knowledge” acquired during pre-training

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the "task knowledge" that complements its pre-existing "language knowledge" acquired during pre-training

The fine-tuning of a pre-trained LLM looks very similar to the training of a machine learning algorithm

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the "task knowledge" that complements its pre-existing "language knowledge" acquired during pre-training

The fine-tuning of a pre-trained LLM looks very similar to the training of a machine learning algorithm

- Select the LLM you wish to use

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the "task knowledge" that complements its pre-existing "language knowledge" acquired during pre-training

The fine-tuning of a pre-trained LLM looks very similar to the training of a machine learning algorithm

- Select the LLM you wish to use
- Tokenise your texts (Note: here we get word-embeddings) and aggregate

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the "task knowledge" that complements its pre-existing "language knowledge" acquired during pre-training

The fine-tuning of a pre-trained LLM looks very similar to the training of a machine learning algorithm

- Select the LLM you wish to use
- Tokenise your texts (Note: here we get word-embeddings) and aggregate
- Perform train-test split (or train-test-validation split for tuning hyperparameters)

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the "task knowledge" that complements its pre-existing "language knowledge" acquired during pre-training

The fine-tuning of a pre-trained LLM looks very similar to the training of a machine learning algorithm

- Select the LLM you wish to use
- Tokenise your texts (Note: here we get word-embeddings) and aggregate
- Perform train-test split (or train-test-validation split for tuning hyperparameters)
- Fine-tune the model on your training data to adapt embeddings and learn task knowledge

Transformer-based models for classification

Although models based on the transformers architecture (like BERT) are primarily trained to predict masked words, they can be further trained to perform tasks such as classification

The training process (techincally called fine-tuning) allows the model to acquire the "task knowledge" that complements its pre-existing "language knowledge" acquired during pre-training

The fine-tuning of a pre-trained LLM looks very similar to the training of a machine learning algorithm

- Select the LLM you wish to use
- Tokenise your texts (Note: here we get word-embeddings) and aggregate
- Perform train-test split (or train-test-validation split for tuning hyperparameters)
- Fine-tune the model on your training data to adapt embeddings and learn task knowledge
- Use fine-tuned model to make predictions on new texts

Let's go back to Google Colab

- 4. Fine-tuning BERT

Natural Language Inference

Natural Language Inference (NLI)

Natural Language Inference (NLI) is a specific type of data format and text classification task, and it's just a bit more complex and nuanced than traditional classification [Laurer et al., 2024]

NLI is the task of determining the relationship between two sentences: a premise and a hypothesis

Natural Language Inference (NLI)

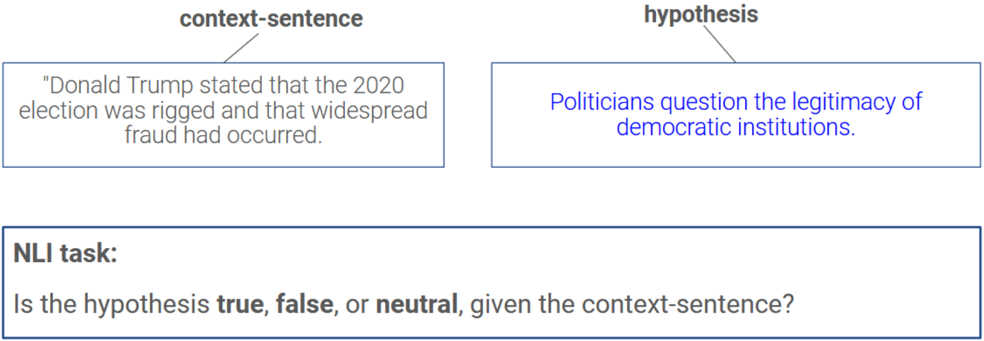
Natural Language Inference (NLI) is a specific type of data format and text classification task, and it's just a bit more complex and nuanced than traditional classification [Laurer et al., 2024]

NLI is the task of determining the relationship between two sentences: a premise and a hypothesis

The model must classify the relationship as one of:

1. Entailment: the hypothesis must be true given the premise
2. Contradiction: the hypothesis must be false given the premise
3. Neutral: the hypothesis could be true or false; the premise gives no clue

Natural Language Inference (NLI)



NLI is about determining if the hypothesis is supported, contradicted, or neutral in relation to the context

Natural Language Inference (NLI)

Context-hypothesis pair → **Input**

"Donald Trump stated that the 2020 election was rigged and that widespread fraud had occurred. **[SEP]** Politicians question the legitimacy of democratic institutions.

Prob of 3 universal classes → **Output**

True: 0.81 Neutral: 0.16. False: 0.03.

Natural Language Inference (NLI)

Why bother about NLI? NLI is a universal task, and almost any classification task can be converted into an NLI task [Laurer et al., 2024]

Example Task:
Identifying texts that indicate support for green policies.

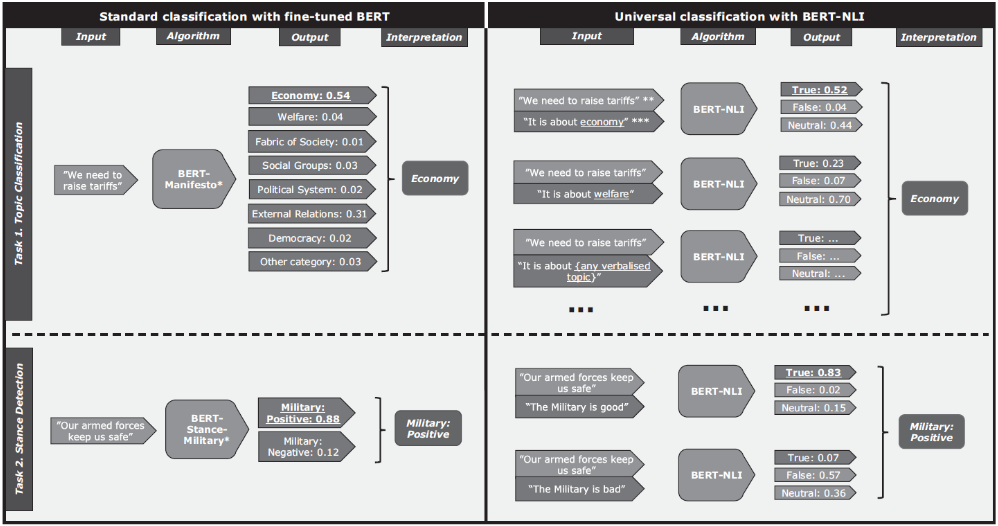
Task Reformulated for NLI:

NLI input	**NLI output**
{context-sentence from your data} [SEP] {hypothesis-sentence verbalising label}	Prob of "True", "False", "Neutral" labels
...	...
The government announced a new plan to reduce carbon emissions by 50% over the next decade. [SEP] The government is supporting green policies.	**True: 0.75** False: 0.10 Neutral: 0.15
The government announced a new plan to reduce carbon emissions by 50% over the next decade. [SEP] The government is opposed to green policies.	True: 0.12 **False: 0.70** Neutral: 0.18

Natural Language Inference (NLI)

NLI input	NLI output
{context-sentence from your data} [SEP] {hypothesis-sentence verbalising label}	Most "True" label
...	**...**
The government increased taxes on the wealthy to fund social programs. [SEP] It is about socialism.	**0.85**
The government increased taxes on the wealthy to fund social programs. [SEP] It is about free-market..	**0.01**
The government increased taxes on the wealthy to fund social programs. [SEP] It is about environmentalism..	**0.18**
The government increased taxes on the wealthy to fund social programs. [SEP] It is about nationalism.	**0.33**

Natural Language Inference (NLI)



Natural Language Inference (NLI)

In the context of LLMs for text classification, NLI has various advantages over other approaches

Natural Language Inference (NLI)

In the context of LLMs for text classification, NLI has various advantages over other approaches

1. Great availability of dataset in NLI-format (hypothesis-context pairs) for training

Natural Language Inference (NLI)

In the context of LLMs for text classification, NLI has various advantages over other approaches

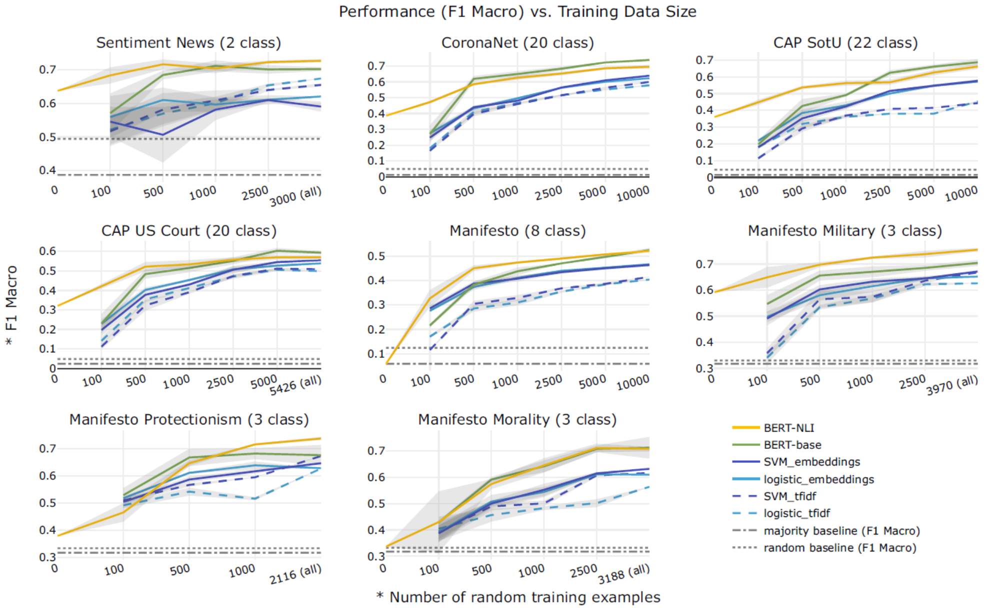
1. Great availability of dataset in NLI-format (hypothesis-context pairs) for training
2. Label verbalisation means that class can be explicitly verbalised in the hypothesis based on a codebook, thus imitating human annotation and allowing the model to build on its prior knowledge

Natural Language Inference (NLI)

In the context of LLMs for text classification, NLI has various advantages over other approaches

1. Great availability of dataset in NLI-format (hypothesis-context pairs) for training
2. Label verbalisation means that class can be explicitly verbalised in the hypothesis based on a codebook, thus imitating human annotation and allowing the model to build on its prior knowledge
3. Performs well even with a small(er) amount of training examples

Natural Language Inference (NLI)



Let's go back to Google Colab

- 5. Fine-tuning BERT-NLI

Classifying texts in multiple languages

Multilingual text classification

Most of the methodologies for text classification are language agnostic; i.e., apart from obvious adjustments, they can work with languages other than English

Multilingual text classification

Most of the methodologies for text classification are language agnostic; i.e., apart from obvious adjustments, they can work with languages other than English

However, challenges can arise if we want to analyse more than one language at the same time. This is what we mean by "multilingual text analysis"

Multilingual text classification

Most of the methodologies for text classification are language agnostic; i.e., apart from obvious adjustments, they can work with languages other than English

However, challenges can arise if we want to analyse more than one language at the same time. This is what we mean by "multilingual text analysis"

Running separate – language-specific models – for each language under analysis hampers the comparability of our findings, unless we "anchor" our models to each other

Multilingual text classification

Approach 1: (Machine) translation

You basically transform a multilingual classification task into a monolingual one, by translating all texts into the same language, and using text analysis methods designed for that language

Multilingual text classification

Approach 1: (Machine) translation

You basically transform a multilingual classification task into a monolingual one, by translating all texts into the same language, and using text analysis methods designed for that language

- This allows you to use unsupervised methods "easily"
- If you are using dictionary, semisupervised or supervised methods, you just need to deal with one language; see Lucas et al. (2015)
- Considerations: are you losing information by translating the texts? How good are the translations? How costly?

Multilingual text classification

Approach 2: Separate but comparable analyses

You keep the original languages, and deal with them individually. However, you ensure comparability of the findings with some preparatory or ex post steps

Multilingual text classification

Approach 2: Separate but comparable analyses

You keep the original languages, and deal with them individually. However, you ensure comparability of the findings with some preparatory or ex post steps

- For unsupervised methods (e.g., LDA or STM), you need to sell that the estimated topics are comparable
- If you are using dictionary, or semisupervised methods, you need to produce comparable sets of keywords; see Maier et al. 2022
- For supervised classification, you have to make sure that the set of training documents is comparable
- Considerations: great multiplication of costs/efforts. Maybe not worth if you have many different languages.

Multilingual text classification

Approach 3: Multilingual word-embeddings

You do not work on tokens directly, but transform them into word-embbeddings that are comparable across languages (same term from different languages will have same embedding).

Multilingual text classification

Approach 3: Multilingual word-embeddings

You do not work on tokens directly, but transform them into word-embbeddings that are comparable across languages (same term from different languages will have same embedding).

- You can use the same classifier across all languages; see Licht 2023
- The machine learning model or LLM will build on information from different languages (and more cases) to make classification
- Considerations: are the embeddings working well in all languages?

Recap

- Word-embeddings represent another way of transforming raw texts into numerical information (vectors)
- As such, they can be used as input in more “traditional” classification methods (like machine learning algorithms)
- Embeddings representation enables models to get a contextualised understanding of tokens (using transformers architecture) and thus to incorporate “language knowledge”
- Transformers-based LLMs can rely on such knowledge to perform universal tasks, or be fine-tuned on more data to acquire “task-knowledge”
- Another way of using LLMs for classification is to adapt a task to a NLI format
- LLMs can be useful alternative in case of mulitlingual classification, as word-embeddings help in mapping texts from different languages into the same semantic space

References I

Laurer, M., Van Atteveldt, W., Casas, A., and Welbers, K. (2024).
Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and bert-nli.
Political Analysis, 32(1):84–100.