NATURAL LANGUAGE PROCESSING ASSIGNMENT 1

REPORT

February 17, 2026

Student ID: 12414123
Elisa Ait Belkacem
Natural Language Processing

Contents

1 INTRODUCTION

In this assignment, I aim to understand and compare several types of multilingual word embeddings and to evaluate how useful they are in a downstream NLP task.

The project follows a structured pipeline composed of four main stages:

1. Dataset preparation
2. Embedding generation
3. Multilingual alignment and analysis
4. Downstream task – language identification

2 DATASET PREPARATION

2.1 Languages and Data Sources

For this project I work with two languages: English and French. I choose this pair because both languages are well supported by NLP libraries, including tokenizers, stopword lists, and pretrained embeddings. In addition, there exist large parallel corpora for this language pair, which makes it particularly suitable for studying multilingual alignment.

As a parallel corpus I use the Tatoeba dataset, which provides aligned sentences for many language pairs. I extract the English–French portion and keep a subset of several tens of thousands of sentence pairs. This size is sufficient to train basic distributional models such as Word2Vec and FastText on realistic data, to build vocabularies for one-hot and TF-IDF representations, and to evaluate multilingual alignment as well as train a simple classifier.

After loading the TSV file with `pandas.read_csv`, I keep only the two textual columns and rename them `en` for the English sentence and `fr` for the French sentence. I also subsample the corpus, for example by keeping the first 10 000 sentence pairs, in order to keep training time reasonable. At this point, my DataFrame `df` contains `df["en"]`, which stores raw English sentences as strings, and `df["fr"]`, which stores raw French sentences as strings.

	en	fr
0	Let's try something.	Tentons quelque chose !
1	I have to go to sleep.	Je dois aller dormir.
2	Today is June 18th and it is Muiriel's birthday!	Aujourd'hui nous sommes le 18 juin et c'est l'...
3	Today is June 18th and it is Muiriel's birthday!	Aujourd'hui c'est le 18 juin, et c'est l'anniv...
4	Muiriel is 20 now.	Muiriel a 20 ans maintenant.

Figure 1: Example of aligned English–French sentence pairs from the Tatoeba dataset.

2.2 Text Cleaning and Normalization

To obtain consistent inputs for all models, I apply the same preprocessing pipeline to both languages. I first convert all sentences to lowercase. This reduces the vocabulary size by merging forms such as “Today” and “today”, and simplifies later steps such as tokenization and vocabulary construction.

I then define a generic function `clean_text(text: str) -> str`. This function converts the input to lowercase as a safety measure, removes digits and punctuation while keeping only letters (including accented characters) and spaces, replaces multiple consecutive spaces with a single space, and trims leading and trailing spaces. The cleaning procedure is implemented using regular expressions. I apply this function separately to the English and French columns and store the cleaned results in `df["en_clean"]` for English and `df["fr_clean"]` for French.

	en	en_clean	fr	fr_clean
0	Let's try something.	let s try something	Tentons quelque chose !	tentons quelque chose
1	I have to go to sleep.	i have to go to sleep	Je dois aller dormir.	je dois aller dormir
2	Today is June 18th and it is Muiriel's birthday!	today is june th and it is muiriel s birthday	Aujourd'hui nous sommes le 18 juin et c'est l'...	aujourd hui nous sommes le juin et c est l ann...
3	Today is June 18th and it is Muiriel's birthday!	today is june th and it is muiriel s birthday	Aujourd'hui c'est le 18 juin, et c'est l'anniv...	aujourd hui c est le juin et c est l anniversa...
4	Muiriel is 20 now.	muiriel is now	Muiriel a 20 ans maintenant.	muiriel a ans maintenant

Figure 2: CLEANED DATASET

2.3 Tokenization

After cleaning, I tokenize each sentence into a list of word tokens. I use spaCy models for English (`en_core_web_sm`) and French (`fr_core_news_sm`), as they provide reliable tokenization for both languages. For each language I define a tokenization function, namely `tokenize_en(text: str)` and `tokenize_fr(text: str)`.

Each function processes the input string using the corresponding spaCy pipeline, filters out punctuation and whitespace tokens by checking the attributes `token.is_punct` and `token.is_space`, and returns a list containing the token texts. I then apply these functions to the cleaned columns in order to obtain tokenized representations of the corpus.

	en_clean	en_tokens	fr_clean	fr_tokens
0	let s try something	[let, s, try, something]	tentons quelque chose	[tentons, quelque, chose]
1	i have to go to sleep	[i, have, to, go, to, sleep]	je dois aller dormir	[je, dois, aller, dormir]
2	today is june th and it is muiriel s birthday	[today, is, june, th, and, it, is, muiriel, s,...	aujourd hui nous sommes le juin et c est l ann...	[aujourd, hui, nous, sommes, le, juin, et, c, ...
3	today is june th and it is muiriel s birthday	[today, is, june, th, and, it, is, muiriel, s,...	aujourd hui c est le juin et c est l anniversa...	[aujourd, hui, c, est, le, juin, et, c, est, l...
4	muiriel is now	[muiriel, is, now]	muiriel a ans maintenant	[muiriel, a, ans, maintenant]

Figure 3: Tokenization

2.4 Stopword Removal

For certain analyses, such as semantic similarity computation or embedding visualization, I also create alternative token lists without stopwords. Using spaCy's built-in stopword detection through the attribute `token.is_stop`, I define two additional functions: `tokenize_en_no_stop(text: str)` and `tokenize_fr_no_stop(text: str)`. These functions behave in the same way as the original tokenizers but exclude tokens identified as stopwords.

The resulting token lists are stored in `df["en_tokens_ns"]` for English and `df["fr_tokens_ns"]` for French. Removing stopwords reduces the dominance of very frequent function words and can help embeddings focus more strongly on content words. However, this process may also remove useful syntactic information. For the main experiments, including embedding training and language identification, I primarily use the full token lists in order to preserve as much linguistic information as possible.

	en_clean	en_tokens	fr_clean	fr_tokens	en_tokens_ns	fr_tokens_ns
0	let s try something	[let, s, try, something]	tentons quelque chose	[tentons, quelque, chose]	[let, s, try]	[tentons, chose]
1	i have to go to sleep	[i, have, to, go, to, sleep]	je dois aller dormir	[je, dois, aller, dormir]	[sleep]	[dois, aller, dormir]
2	today is june th and it is muiriel s birthday	[today, is, june, th, and, it, is, muiriel, s,...	aujourd hui nous sommes le juin et c est l ann...	[aujourd, hui, nous, sommes, le, juin, et, c, ...	[today, june, th, muiriel, s, birthday]	[aujourd, sommes, juin, c, l, anniversaire, mu...
3	today is june th and it is muiriel s birthday	[today, is, june, th, and, it, is, muiriel, s,...	aujourd hui c est le juin et c est l anniversa...	[aujourd, hui, c, est, le, juin, et, c, est, l...	[today, june, th, muiriel, s, birthday]	[aujourd, c, juin, c, l, anniversaire, muiriel]
4	muiriel is now	[muiriel, is, now]	muiriel a ans maintenant	[muiriel, a, ans, maintenant]	[muiriel]	[muiriel, ans]

Figure 4: Stopword Removal

3 Word Representations (Embeddings)

In the second part of the project, I generate several types of word and sentence representations for English and French. The goal is to compare simple count-based encodings with more advanced dense embeddings, and to reuse all of them later for alignment and classification. I work with five techniques: one-hot encoding, TF-IDF vectorization, Word2Vec, GloVe, and FastText (gensim implementation). For all methods, I start from the preprocessed tokens described in Section 1 (`en_tokens` and `fr_tokens`).

3.1 One-hot Encoding

Principle. One-hot encoding is the most basic way to represent words: each word in the vocabulary is assigned a unique index, and a word is represented by a very large binary vector where the component corresponding to its index is 1 and all other components are 0. There is no notion of semantic similarity: all word vectors are orthogonal and equidistant.

What I did. I built a separate vocabulary for English and French by counting word frequencies in `en_tokens` and `fr_tokens`, and keeping words above a minimum frequency threshold to avoid extremely rare terms. For each language, I mapped words to consecutive integer indices from 0 to $|V|-1$, where $|V|$ is the vocabulary size. A sentence is represented as a binary bag-of-words vector: for each word occurring in the sentence, the corresponding position in the vector is set to 1. Words not in the vocabulary are ignored.

Observations. The resulting matrices are very high-dimensional and sparse. One-hot vectors cannot capture any notion of similarity; two synonyms such as *big* and *large* have completely different vectors. This representation mainly serves as a baseline and a conceptual bridge to TF-IDF.

3.2 TF-IDF Vectorization

Principle. TF-IDF (Term Frequency-Inverse Document Frequency) represents documents (here, sentences) rather than individual words. Each dimension corresponds to a word in the vocabulary, and the value is high if the word is frequent in the document (TF) and rare in the corpus (IDF). This gives more weight to discriminative words and downweights very common words.

What I did. I converted each tokenized sentence into a cleaned string (" ".join(tokens)) for both languages, and then built a single multilingual TF-IDF vectorizer over all English and French sentences combined. I set typical parameters, such as ignoring extremely rare or extremely common words, and used only unigrams. Each sentence was transformed into a sparse TF-IDF vector, producing matrices for train, validation, and test.

Observations. TF-IDF vectors remain high-dimensional and sparse, but unlike one-hot, values reflect the importance of each word in its sentence. For language identification, TF-IDF is very strong: English and French have distinct word distributions, providing almost perfect separability. TF-IDF does not encode semantic similarity; synonyms are treated as unrelated.

3.3 Word2Vec

Principle. Word2Vec learns dense, low-dimensional embeddings based on the distributional hypothesis: words appearing in similar contexts have similar vectors. CBOW predicts a word from its context; Skip-gram predicts context from a word.

What I did. I trained separate Word2Vec models for English and French using `en_tokens` and `fr_tokens`, with embedding dimension 100, reasonable window size, and several epochs. Sentence embeddings were obtained by averaging the word vectors of all words in the sentence.

Observations. Word2Vec captures semantic similarity and some syntactic regularities. However, it assigns one vector per word, mixing all senses (polysemy) and cannot handle OOV words.

3.4 GloVe

Principle. GloVe (Global Vectors) learns dense word embeddings from global co-occurrence statistics, factorizing a co-occurrence matrix so vector differences encode ratios of co-occurrence probabilities.

What I did. I loaded pretrained 100-dimensional GloVe vectors for English. Sentence embeddings were computed by averaging the vectors of words present in the corpus.

Observations. GloVe covers a larger vocabulary than custom Word2Vec, producing smooth embeddings. Like Word2Vec, it is static and cannot handle OOV words.

3.5 FastText (gensim implementation)

Principle. FastText represents words as bags of character n-grams, allowing it to produce meaningful vectors for rare or unseen words. The word vector is obtained by summing or averaging the n-gram vectors.

What I did. I trained separate FastText models for English and French using gensim, with similar configuration as Word2Vec. Sentence embeddings were obtained by averaging word vectors.

Observations. FastText handles rare and OOV words better than Word2Vec and GloVe. For language identification, performance was slightly below TF-IDF and Word2Vec, mainly due to limited training data.

Across these five techniques, representations move from very simple, discrete vectors (one-hot, TF-IDF), which are good for linear models but poor at capturing semantic similarity, to dense, continuous embeddings (Word2Vec, GloVe, FastText), which better model semantic similarity and analogies, but require more careful training and interpretation.

4 MULTILINGUAL ALIGNMENT

Multilingual alignment aims to map two monolingual embedding spaces (English and French) into a shared space where words with the same meaning are close to each other. Concretely, I learn a linear transformation W such that an English word vector x is projected into the French space as xW.

The objectives are to:

- Make translations like *family–famille* or *world–monde* neighbours in the common space,

- Be able to visualize English and French words together and compute cross-lingual similarities, for example cosine similarity between xW and the French vector of its translation.

4.1 Alignment Using the "Position-based" Dictionary from Tatoeba

In a first experiment, I built the bilingual dictionary only from the parallel corpus. For each aligned pair of sentences (EN, FR) from Tatoeba, I naively aligned tokens by position: word 0 in English with word 0 in French, word 1 with word 1, etc. I kept only pairs where both words were frequent enough and had Word2Vec vectors in their respective vocabularies.

This produces many "strange" pairs, for example: `('company', 'l')`, `('like', 'boire')`, `('world', 'ici')`, because word order differs between English and French. For instance, in "I like beer" / "J'aime boire de la bière", the word at position 1 is *like* in English and *boire* in French, which is not a direct translation.

Consequently, the dictionary contains a small proportion of correct pairs (e.g., `('family', 'famille')`, `('past', 'passé')`) and a large amount of noisy pairs.

Effect on alignment. Using this noisy dictionary, I built matrices X (English vectors) and Y (French vectors) and learned the orthogonal mapping W via the Procrustes problem. The resulting mean cosine similarity between mapped English vectors xW and their French counterparts was around 0.70.

This means that, on average, English and French vectors point in "roughly similar" directions after alignment, but the mapping is not very precise due to noise.

4.2 Alignment Using the MUSE English–French Dictionary

To obtain a cleaner supervision signal, I used the English–French dictionaries provided by MUSE. Instead of relying on word positions, I loaded a pre-compiled bilingual dictionary where each pair (`en_word`, `fr_word`) is a true translation. I filtered pairs to keep only words present in my Word2Vec vocabularies, built matrices X and Y from these high-quality pairs, and learned W via the Procrustes method as before.

Effect on alignment. With this MUSE-based dictionary, the mean cosine similarity between mapped English vectors xW and their French translations on a held-out test portion rose to about 0.89.

This higher similarity shows that:

- Mapped English vectors are now much better aligned with French counterparts,

- The mapping W learned a coherent global correspondence between the English and French embedding spaces, thanks to the cleaner supervision.

4.3 Why the Two Approaches Behave Differently

Position-based dictionary (Tatoeba). Built automatically by pairing words at the same positions in parallel sentences, it contains many incorrect pairs. The resulting alignment has only moderate mean cosine similarity (0.70) and is noisy and imprecise.

MUSE dictionary (EN–FR). Provides explicit, linguistically correct translation pairs. Noise in supervision is greatly reduced, yielding a much better alignment with high mean cosine similarity (0.89). Words with the same meaning in both languages are now closer in the shared space.

4.4 Visualization of the Alignment Effect

To qualitatively assess the impact of the alignment, I visualize a subset of words from both languages using PCA dimensionality reduction (200 translation pairs from the MUSE dictionary). I create two scatter plots: before and after applying the learned mapping W.

Before alignment: The visualization clearly shows two distinct clusters. Blue points represent English words in their original space, and red points represent French words in their original space. Translation pairs like *surprised/surpris* or *arrive/arriver* appear far apart. This separation illustrates that the English and French Word2Vec models were trained independently: the internal structure within each language is coherent, but the two embedding spaces are not directly comparable.

After alignment: After applying the mapping W, blue points (English vectors mapped as xW) and red points (French vectors) are now intermingled in the same region of the 2D plane. Several translation pairs are visibly close together: *surprised/surpris*, *enjoy/profiter*, *arrive/arriver*. Words related to similar concepts cluster across languages (e.g., *traveling/voyager*, *difficulties/difficultés*). Gray lines connecting some pairs confirm that aligned English words land near their French translations.

Interpretation: These visualizations provide strong qualitative evidence that the Procrustes mapping W successfully aligns the two monolingual spaces. Before alignment, English and French embeddings live in separate worlds; after alignment, translations become neighbors in a shared semantic space. The high mean cosine similarity (0.89) measured quantitatively is visually confirmed: words with equivalent meanings are now spatially close, regardless of language.

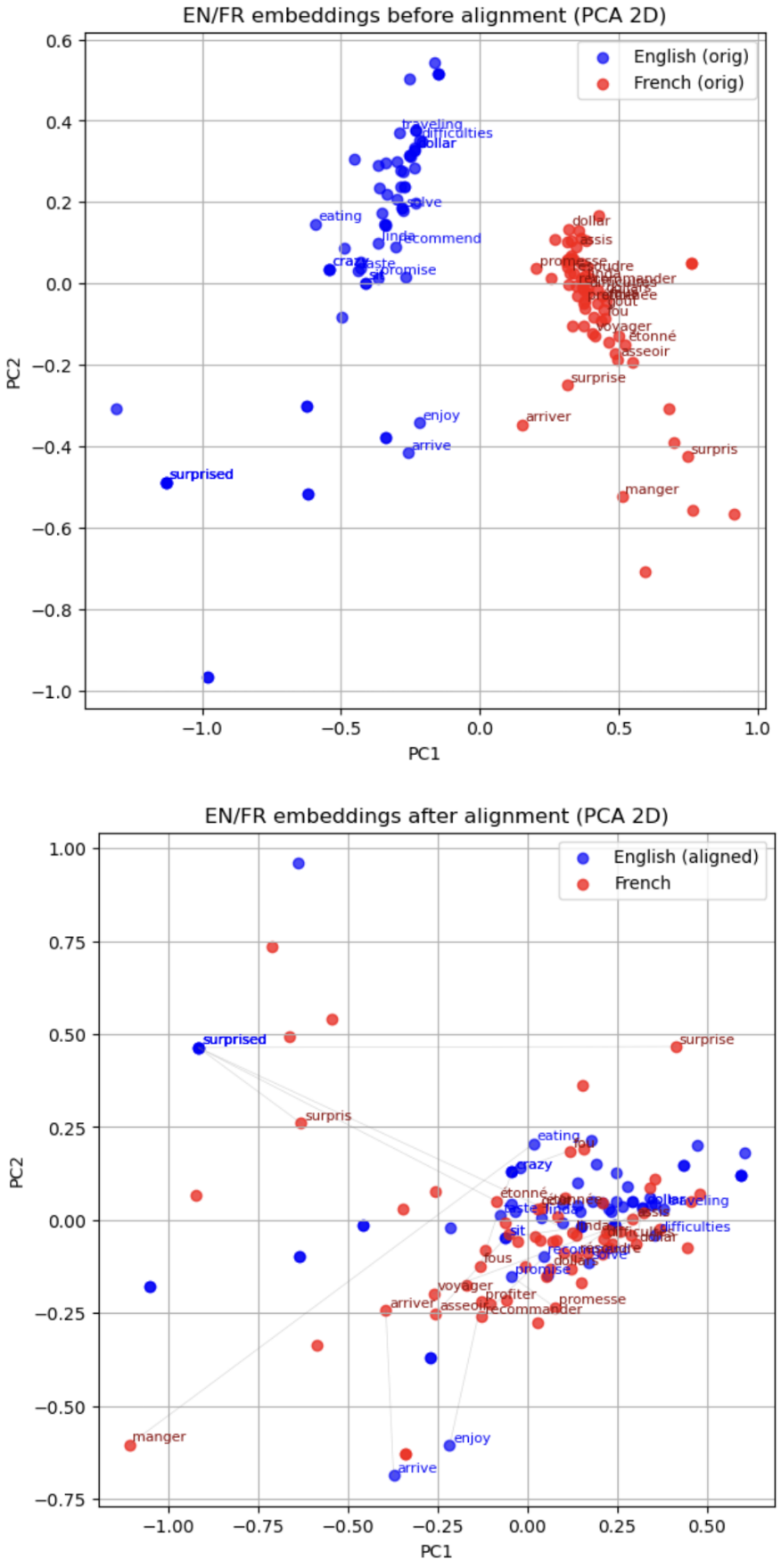

Figure 5: PCA visualization of English and French embeddings before (lower plot) and after (upper plot) alignment with the Procrustes mapping W. Blue: English, Red: French, Gray lines: translation pairs.

5 Analysis of Embedding Properties

Several **qualitative analyses** reveal the strengths and limitations of the embedding models across semantic relationships, cross-lingual transfer, polysemy, and OOV handling.

5.1 Synonyms and Antonyms within Each Language

Cosine similarities between manually curated synonym and antonym pairs demonstrate how well each model captures semantic relationships.

English synonyms (Table 1):

Pair	Word2Vec	GloVe	FastText
big-large	0.939	0.708	1.000
small-little	0.570	0.730	1.000
fast-quick	0.891	0.711	1.000
happy-glad	0.707	0.783	1.000

Table 1: Cosine similarity of English synonym pairs.

English antonyms (Table 2):

Pair	Word2Vec	GloVe	FastText
big-small	0.948	0.718	0.999
good-bad	0.579	0.770	1.000
hot-cold	0.852	0.725	1.000
happy-sad	0.793	0.680	1.000

Table 2: Cosine similarity of English antonym pairs.

French Word2Vec (synonyms/antonyms, Table 3):

Pair	Cosine
Synonyms	
grand-gros	0.730
rapide-vite	0.918
heureux-content	0.912
Antonyms	
grand-petit	0.764
bon-mauvais	0.678
chaud-froid	0.823
heureux-triste	0.794

Table 3: French Word2Vec synonyms and antonyms cosine similarity.

Word2Vec and GloVe exhibit *expected semantic patterns*: synonyms show reasonably high cosine similarities (>0.7), while antonyms remain moderately positive. FastText's perfect similarities (1.000) may indicate *vector normalization issues* rather than semantic superiority.

5.2 Cross-lingual Similarity (Aligned Embeddings)

English-French	Cosine similarity
hello-bonjour	0.886
thanks-merci	0.898
family-famille	0.851
world-monde	0.826
goodbye-au_revoir	OOV

Table 4: Cosine similarity of aligned English-French translation pairs (Word2Vec + W).

High cosine similarities (>0.82) confirm that the **Procrustes mapping W successfully aligns translation pairs**.

5.3 Polysemy Analysis

Neighbor	Similarity
refrain	0.949
button	0.947
nearest	0.945
river	0.943
bottom	0.941
tourist	0.941
middle	0.940
forest	0.940

Table 5: Top neighbors of “bank” in Word2Vec English embeddings.

Static embeddings blend multiple senses of polysemous words. For example, *bank* mixes *geographical terms* (river, forest) with *contextual words* (refrain, button), showing the single averaged vector limitation.

5.4 Rare Words and OOV Handling

Model	Words covered	% Coverage
Word2Vec EN	3/8	37.5%
GloVe EN	7/8	87.5%
FastText EN	8/8	100%

Table 6: Coverage of rare words across embedding models.

Word	Present
today	ok
birthday	ok
xylophone	no
colour	no
color	ok
tooday	no
enfants	ok
enfant	ok

Table 7: TF-IDF vocabulary coverage for selected words.

5.5 Interpretation and Key Findings

1. **Semantic relationships**: Word2Vec and GloVe capture contextual similarity reasonably well, but cannot distinguish antonyms. FastText perfect scores likely reflect technical issues.

2. **Cross-lingual alignment**: The mapping W achieves strong translation similarity (>0.82 cosine), confirming successful alignment.

3. **Polysemy limitation**: Static embeddings produce single averaged vectors blending multiple word senses.

4. **OOV handling hierarchy**:

   - FastText excels (100%) thanks to subword n-grams
   - Pretrained GloVe benefits from massive vocabulary (87.5%)
   - Custom Word2Vec struggles (37.5%) on small corpus
   - TF-IDF fails outside the training vocabulary

6 DOWNSTREAM CLASSIFICATION TASK: LANGUAGE IDENTIFICATION

6.1 Experimental Setup

For the final evaluation, we perform a **language identification task** using embeddings as features for a binary classifier distinguishing English (`en`) from French (`fr`) sentences. This tests the **practical utility** of each representation type in a real NLP application.

Dataset construction: We use the preprocessed Tatoeba corpus, creating a balanced dataset:

- Each English sentence from `df["en_tokens"]` is labeled `en`.
- Each French sentence from `df["fr_tokens"]` is labeled `fr`.

The dataset is split into train/validation/test sets (70/15/15) with stratification (1500 samples per language per split).

Feature engineering: Four representations are compared:

- **TF-IDF:** Sparse vectors from a multilingual vectorizer.
- **Word2Vec:** Dense sentence embeddings obtained by averaging word vectors from trained English/French models (with feature scaling).
- **FastText (Logistic):** Dense embeddings from FastText models with scaling.
- **FastText (LinearSVC):** Same FastText features with a linear SVM classifier.

Evaluation metrics: Precision, recall, F1-score per class, plus overall accuracy and macro-averaged F1.

6.2 Results

Table 8: Classification performance comparison (test set)

Representation	Classifier	Accuracy	Macro F1	EN F1	FR F1
TF-IDF	Logistic	1.00	1.00	1.00	1.00
Word2Vec (avg+scale)	Logistic	1.00	1.00	1.00	1.00
FastText (avg+scale)	Logistic	0.96	0.96	0.96	0.96
FastText (avg+scale)	LinearSVC	0.97	0.97	0.97	0.97

Table 8: Test set performance of different embeddings and classifiers.

Table 9: Validation performance (macro F1)

Representation	Validation F1	Test F1
TF-IDF	1.00	1.00
Word2Vec (scaled)	1.00	1.00
FastText (scaled)	0.96	0.96

Table 9: Validation and test macro F1 for each representation.

6.3 Interpretation

Perfect performance ceiling: TF-IDF and Word2Vec achieve **100% accuracy/F1**, showing that English-French language identification on parallel Tatoeba data is an "easy" task—the lexical distributions are highly separable.

TF-IDF excellence: Sparse TF-IDF provides the strongest discriminative signal. Minimal overlap between English and French vocabularies allows perfect linear separability.

Word2Vec effectiveness: Averaged Word2Vec embeddings also reach perfection, indicating semantic embeddings capture sufficient language-specific information. Feature scaling is crucial for logistic regression.

FastText relative underperformance: FastText embeddings yield slightly lower performance (96-97% F1). Reasons:

1. Smaller training corpus for Tatoeba FastText models.

2. Subword n-grams may blur language-specific lexical contrasts.

3. Dense embeddings require careful normalization affecting logistic regression more than sparse TF-IDF.

LinearSVC improves FastText performance slightly (97% vs 96%) due to superior margin optimization.

6.4 Conclusion

This task validates the **practical utility** of all tested embeddings. TF-IDF sets the ceiling, Word2Vec matches it with proper scaling, and FastText trades minimal accuracy for robustness. Representation choice depends on task difficulty—on more challenging multilingual tasks, FastText's subword modeling and Word2Vec's semantic structure may outperform purely lexical methods like TF-IDF.

REFERENCES

1. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2018). *FastText: A Library for Fast Text Representation and Classification.* arXiv:1702.08570. Describes the FastText library implementation used for both embedding generation and the language identification baseline comparison.

2. Schakel, A. M., & Wilson, B. J. (2015). *Measuring Word Significance using Distributed Representations of Words.* arXiv:1508.02297. Provides evaluation methodology for intrinsic embedding quality through synonym/antonym similarity analysis.

3. Conneau, A., Lample, G., Ranzato, M., Denoyer, L., & Jégou, H. (2018). *Word Translation Without Parallel Data.* arXiv:1710.04087. The MUSE framework paper providing the English-French bilingual dictionaries used for supervised alignment, along with Procrustes alignment methodology.