PROGRAMME DE FORMATION

TRAITEMENT AUTOMATIQUE DU LANGAGE NATUREL (NLP) AVEC PYTHON

FICHE SYNTHÉTIQUE DE LA FORMATION

Intitulé	Traitement Automatique du Langage Naturel (NLP) avec Python
Durée	4 jours (28 heures)
Référence	PY-NLP-4J
Tarif	2 500 € HT / personne
Modalité	Présentiel ou distanciel
Public	Développeurs Python, data scientists, ingénieurs ML, chercheurs
Prérequis	Maîtrise de Python (POO, modules). Notions de base en mathématiques (lycée scientifique).
Lieu	255 boulevard Pereire, 75017 Paris ou à distance (Zoom)

1. Public visé

Cette formation s'adresse aux développeurs Python, data scientists, ingénieurs en machine learning et chercheurs souhaitant acquérir une maîtrise solide du traitement automatique du langage naturel. Elle couvre à la fois les fondations mathématiques nécessaires, les techniques classiques du NLP et les approches modernes basées sur le deep learning avec PyTorch et TensorFlow.

2. Prérequis

- Maîtrise de Python niveau intermédiaire : POO, modules, manipulation de fichiers
- Notions de base en mathématiques : opérations vectorielles, fonctions, dérivées (niveau lycée scientifique)
- Connaissance de pandas et numpy appréciée mais non obligatoire
- Disposer d'un ordinateur avec Python 3.x et accès internet

Un test de positionnement est proposé avant l'entrée en formation afin de vérifier l'adéquation du parcours avec le profil du stagiaire.

3. Objectifs pédagogiques

À l'issue de cette formation, le stagiaire sera capable de :

• Maîtriser les concepts d'algèbre linéaire essentiels au NLP (vecteurs, matrices, produits scalaires, projections)

• Mettre en œuvre les techniques classiques de prétraitement du texte (tokenization, stemming, lemmatisation)

• Construire des représentations vectorielles de mots (TF-IDF, Word2Vec, GloVe, embeddings contextuels)

• Développer des modèles de classification de texte avec scikit-learn

• Concevoir et entraîner des réseaux de neurones pour le NLP avec PyTorch

• Construire des modèles séquentiels (RNN, LSTM, GRU) avec TensorFlow/Keras

• Comprendre l'architecture des Transformers et utiliser Hugging Face

• Développer une application NLP complète (analyse de sentiment, NER, résumé automatique)

4. PROGRAMME DÉTAILLÉ

JOUR 1 — Fondations mathématiques et premiers pas en NLP (7h)

Matin (3h30) – Algèbre linéaire pour le NLP

• Vecteurs : définition, dimension, représentation d'un mot ou d'un document

• Opérations vectorielles : addition, multiplication scalaire, norme (L1, L2)

• Produit scalaire et similarité cosinus : mesurer la proximité sémantique entre textes

• Matrices : définition, dimensions, transposition, produit matriciel

• Matrices et transformations linéaires : projection, rotation, changement de base

• Décomposition en valeurs singulières (SVD) : principe et application au NLP (LSA)

• Calcul matriciel avec NumPy : ndarray, broadcasting, opérations vectorisées

• Notions de probabilités utiles : distributions, probabilité conditionnelle, théorème de Bayes

Méthodes : Démonstrations visuelles, exercices guidés sur NumPy

Après-midi (3h30) – Prétraitement du texte et NLP classique

• Panorama du NLP : enjeux, applications, évolution historique

• Tokenization : mots, sous-mots, phrases — bibliothèques NLTK et spaCy

• Normalisation : minuscules, accents, ponctuation, stop words

• Stemming et lemmatisation : différences et cas d'usage

• POS tagging : étiquetage grammatical avec spaCy

• Reconnaissance d'entités nommées (NER) : personnes, lieux, organisations

• Représentations Bag-of-Words et TF-IDF : principe et calcul

Méthodes : Manipulation interactive de corpus, exercices progressifs

Travaux pratiques :

• TP1 : Calcul de similarité cosinus entre documents avec NumPy

• TP2 : Pipeline complet de prétraitement de texte avec spaCy (tokenization, lemmatisation, NER)

• TP3 : Construction d'une matrice TF-IDF et recherche des documents les plus similaires

Évaluation formative : Correction collective des TP, tour de table de fin de journée

JOUR 2 — Embeddings et machine learning classique appliqué au NLP (7h)

Matin (3h30) – Word embeddings

• Limites des représentations Bag-of-Words : sparsité, absence de sémantique
• Hypothèse distributionnelle : « un mot se définit par son contexte »
• Word2Vec : architectures CBOW et Skip-Gram, intuition mathématique
• GloVe : factorisation de matrice de co-occurrence
• FastText : embeddings de sous-mots, gestion des mots inconnus
• Visualisation des embeddings : réduction de dimension avec PCA et t-SNE
• Utilisation de modèles pré-entraînés (français et anglais)

Méthodes : Démonstrations visuelles, exploration interactive d'embeddings

Après-midi (3h30) – Classification de texte avec scikit-learn

• Pipeline de machine learning pour le NLP : prétraitement, vectorisation, modèle, évaluation
• Algorithmes classiques : régression logistique, SVM, Naive Bayes, Random Forest
• Métriques d'évaluation : précision, rappel, F1-score, matrice de confusion
• Cross-validation et recherche d'hyperparamètres avec GridSearchCV
• Gestion des classes déséquilibrées : oversampling, class weights
• Interprétabilité des modèles : importance des mots, LIME

Méthodes : Exercices guidés sur des datasets réels

Travaux pratiques :

• TP1 : Entraînement de Word2Vec sur un corpus français et exploration des analogies
• TP2 : Classifieur de sentiment sur des avis clients avec scikit-learn (TF-IDF + régression logistique)
• TP3 : Détection automatique de spam avec optimisation d'hyperparamètres

Évaluation formative : QCM mi-parcours (20 questions), correction des TP en groupe

JOUR 3 — Deep learning pour le NLP avec PyTorch (7h)

Matin (3h30) – Fondamentaux de PyTorch

• Présentation de PyTorch : philosophie, écosystème, comparaison avec TensorFlow
• Tenseurs PyTorch : création, opérations, broadcasting, transfert CPU/GPU
• Autograd : différentiation automatique, calcul du gradient
• Construction d'un réseau de neurones : torch.nn.Module, couches Linear, fonctions d'activation
• Boucle d'entraînement : forward, loss, backward, optimizer (SGD, Adam)
• DataLoader et Dataset : chargement efficace des données textuelles

- Sauvegarde et chargement de modèles

Méthodes : Démonstrations live, construction pas à pas d'un premier modèle

Après-midi (3h30) – Réseaux de neurones pour le NLP avec PyTorch

- Couche Embedding : représentation dense des mots dans un réseau de neurones
- Réseaux feed-forward pour la classification de texte
- Réseaux récurrents (RNN) : intuition, équations, limites (vanishing gradient)
- LSTM et GRU : mécanismes de portes, mémoire à long terme
- RNN bidirectionnels : combiner contexte gauche et droite
- Régularisation : dropout, batch normalization, early stopping
- Évaluation et débogage d'un modèle de deep learning

Méthodes : Exercices progressifs, construction itérative de modèles

Travaux pratiques :

- TP1 : Implémentation d'un classifieur de texte avec un réseau feed-forward en PyTorch
- TP2 : Construction d'un LSTM pour la classification d'avis clients
- TP3 : Comparaison des performances entre régression logistique, LSTM et GRU sur un même dataset

Évaluation formative : Correction collective des TP, tour de table de fin de journée

JOUR 4 — TensorFlow, Transformers et projet final (7h)

Matin (3h30) – TensorFlow / Keras et architecture Transformer

- Présentation de TensorFlow et Keras : philosophie, API séquentielle et fonctionnelle
- Construction d'un modèle Keras : Sequential, couches, compilation, entraînement
- Tenseurs TensorFlow : tf.Tensor, opérations, eager execution
- Pipeline tf.data : chargement et transformation efficaces de corpus textuels
- Implémentation d'un LSTM en Keras et comparaison avec PyTorch
- Mécanisme d'attention : intuition, calcul, intérêt
- Architecture Transformer : self-attention, multi-head attention, encodage positionnel
- BERT, GPT et leurs variantes : présentation et cas d'usage

Méthodes : Démonstrations comparatives PyTorch / TensorFlow

Après-midi (3h30) – Hugging Face, fine-tuning et projet final

- Écosystème Hugging Face : Transformers, Datasets, Tokenizers, Hub
- Chargement et utilisation de modèles pré-entraînés (CamemBERT, BERT, DistilBERT)
- Pipeline Hugging Face : classification, NER, question-answering, résumé
- Fine-tuning d'un modèle pré-entraîné sur un dataset métier
- Compatibilité PyTorch et TensorFlow dans Hugging Face
- Bonnes pratiques : gestion de la mémoire, GPU, taille de batch

• Projet final : développement d'une application NLP complète

Méthodes : Mini-projet intégrateur en autonomie guidée

Travaux pratiques :

• TP1 : Implémentation et entraînement d'un LSTM en Keras/TensorFlow sur un corpus textuel

• TP2 : Fine-tuning de CamemBERT pour la classification de sentiment sur des avis en français

• TP3 (Projet final) : Application NLP complète au choix — analyse de sentiment, NER métier, ou résumé automatique

Évaluation formative : Soutenance du mini-projet, QCM final de 30 questions

5. MÉTHODES ET MOYENS PÉDAGOGIQUES

MÉTHODES MOBILISÉES

• Apports théoriques : exposés interactifs avec diaporama, démonstrations mathématiques visuelles

• Mise en pratique : TP individuels, exercices progressifs, mini-projet intégrateur

• Pédagogie active : résolution de problèmes collaboratifs, débogage en groupe

• Alternance théorie/pratique : ratio 35 % théorie / 65 % pratique

• Comparaison des frameworks : chaque concept est illustré en PyTorch et TensorFlow lorsque pertinent

MOYENS TECHNIQUES

• Salle de formation équipée : vidéoprojecteur, paperboard, connexion Wi-Fi haut débit

• Poste informatique par stagiaire avec Python, NumPy, scikit-learn, PyTorch, TensorFlow et Hugging Face pré-installés

• Accès à des notebooks Jupyter et à des GPU (en local ou via Google Colab)

• En distanciel : Zoom avec partage d'écran, tableau blanc collaboratif

• Supports de cours et notebooks remis aux stagiaires

6. MODALITÉS D'ÉVALUATION

Avant l'entrée en formation, chaque participant complète un test de positionnement évaluant son niveau en Python, en mathématiques et en machine learning.

POSITIONNEMENT À L'ENTRÉE

• Test de positionnement en ligne avant la formation

ÉVALUATIONS EN COURS DE FORMATION

• Évaluation formative : QCM mi-parcours de 20 questions (Jour 2)

• Évaluation sommative : QCM final de 30 questions (Jour 4)

• Critères de réussite : obtenir 60 % au QCM final

• Attestation de fin de formation : remise à l'issue de la formation, mentionnant les objectifs, la durée et les résultats

SATISFACTION

• Questionnaire de satisfaction en fin de formation (formateur, contenu, logistique, atteinte des objectifs)

7. ACCESSIBILITÉ AUX PERSONNES EN SITUATION DE HANDICAP

Pythonia s'engage à rendre ses formations accessibles aux personnes en situation de handicap. Notre référent handicap est disponible pour étudier les aménagements nécessaires :

• Adaptation des supports pédagogiques (taille de police, contraste, format audio)

• Aménagement des conditions de passage (temps majoré, pauses adaptées)

• Orientation vers nos partenaires spécialisés : Agefiph, Cap Emploi, MDPH

Contact référent handicap : contact@pythonia.fr – 06 66 06 38 59

8. ENGAGEMENT ET SUIVI

• Suivi de l'assiduité : feuille d'émargement signée par demi-journée

• Prévention des abandons : suivi individualisé, adaptation du rythme, relance en cas d'absence

• Réclamations : toute réclamation peut être adressée par email à contact@pythonia.fr. Elle sera traitée sous 48h ouvrées.