Rapport PFE

Table des matières

1 Présentation du contexte et organisation d'accueil

1.1 Présentation des objectifs du projet

Dans cette travail on cherche à améliore la performance du modéle GLiNER pour la task NER dans la domaine de Bio-médicale .On établir un protocole d'évaluation précise en " character level " pour l'évaluation du modèle GLiNER de Base et ses différent version . Pour cela on procède à des approches différents d'évaluation servent à capter les lacunes et le point fort de notre modéle et aussi de la qualités et l'efficacité de nos labels afin d'améloire les prédictions de notre modéle . Ainsi , on aller plus loins dans les recherches on proposons des approches d'amélioration de GLiNER .

1.2 Problématique générale de la NER

La Reconnaissance d'Entités Nommées (NER) est une tâche fondamentale en Traitement Automatique du Langage Naturel (TALN) visant à identifier et classer dans un text des segments appelés *entités*, comme les noms de personnes, des maladies, des lieux, des dates, etc. Cette tâche permet de structurer de l'information brute et de l'exploiter dans des applications variées . Dans cette context il y'avait plusieur approches et modéles developée durant les année Depuis son apparition en 1996 . [10]

1.3 Présentation de l'ISIS et l'IRIT

Le stage est réalisé au sein de l'équipe SIG, dans le département Gestion de Données de l'Institut de Recherche en Informatique de Toulouse (IRIT). L'IRIT est un laboratoire de recherche constitué de chercheurs en informatique, développant notamment des travaux autour du traitement automatique du langage, de la gestion des données, des connaissances et de l'extraction d'informations. Le stage sera localisé sur la laboratoire CHL au seins de l'**Institut Supérieur de l'Informatique de la Santé** (ISIS) situé à Castres est une école d'ingénieurs publique, rattachée à l'Institut National Universitaire Champollion. Elle forme des ingénieurs spécialisés en **e-santé**, à l'interface entre l'informatique et les sciences de la santé.

1.4 Lien des organisations ISIS et IRIT avec le domaine de la santé

L'ISIS développe des projets de recherche dans le domaine de la **santé numérique**, incluant la conception d'applications cliniques, des projet d'application de machine learning et IA pour l'analyse et l'exploitation des données médicales. L'école dispose

également d'un laboratoire vivant, le " *Connected Health Lab*", pour prototyper et tester des solutions innovantes en santé.

De son côté, l'IRIT inscrit le domaine *Santé, autonomie, bien-être* parmi ses axes stratégiques. Plusieurs projets de recherche sont orientés vers l'analyse de données cliniques, la reconnaissance d'entités médicales et l'aide à la décision médicale. L'équipe SIG peut, dans ce cadre, apporter son expertise en modélisation des données, en bases de données médicales et en fouille de données pour la santé. [4]

2 Définition des variables et notations

Soit un texte tokenisé de longueur N, et un ensemble de M types d'entités $\{t_1, \ldots, t_M\}$. Dans le protocole et le mécanisme de correspondance span–type, on utilise les notations suivantes :

- $i, j \in \{1, \ldots, N\}$: indices de début et de fin d'un *span* de tokens dans la séquence.
- $K \in \mathbb{N}$: longueur maximale considérée pour un span, alors on ne forme que les couples (i, j) tels que $1 \leq i < j \leq N$ et $j - i \leq K$. Typiquement $K = 12$.
- $h_i \in \mathbb{R}^D$: vecteur de représentation contextuelle du token en position i, produit par l'encodeur bidirectionnel (BERT) de dimension D.
- $S_{ij} \in \mathbb{R}^{D'}$: représentation du span (i, j), obtenue par

$$S_{ij} \;=\; \mathrm{FFN}\Big(h_i \otimes h_j\Big) \tag{1}$$

où FFN est un réseau *feed-forward* à deux couches position-wise(traite chaque position de manière *indépendante* des autres. Concrètement, si l'on note x_i le vecteur de dimension "d" associé à la position i d'une séquence, alors la transformation position-wise s'applique à chacune de ces positions via la même fonction , et $(h_i \otimes h_j)$ la concaténation des vecteurs contextuels $h_i, h_j \in \mathbb{R}^D$. Plus précisément, chaque représentation de span est calculée comme :

$$S_{ij} = W_2\Big(\sigma\Big(W_1\,(h_i \otimes h_j) + b_1\Big)\Big) + b_2$$

avec :

- $W_1 \in \mathbb{R}^{D_{\mathrm{ff}} \times 2D}$ et $b_1 \in \mathbb{R}^{D_{\mathrm{ff}}}$: poids et biais de la première couche linéaire, projetant la concaténation $(h_i \otimes h_j)$ dans un espace intermédiaire de dimension D_{ff}.
- σ : fonction d'activation non-linéaire (typiquement ReLU ou GELU).
- $W_2 \in \mathbb{R}^{D' \times D_{\mathrm{ff}}}$ et $b_2 \in \mathbb{R}^{D'}$: poids et biais de la seconde couche linéaire, ramenant la dimension à D'.
- Le résultat $S_{ij} \in \mathbb{R}^{D'}$ est la *représentation enrichie* du sous-span (i, j),

incorporant à la fois les informations des positions de début et de fin.

— $q_t \in \mathbb{R}^{D'}$: embedding du type d'entité t, initialisé à partir du token spécial [ENT] et projeté dans le même espace de dimension D'.

— $\varphi(i, j, t) \in [0, 1]$: score de compatibilité entre le span (i, j) et le type t, défini par

$$\varphi(i, j, t) \;=\; \sigma\Big(S_{ij}^{\top}\, q_t\Big) \;=\; \frac{1}{1 + \exp\Big(-\, S_{ij}^{\top} q_t\Big)},$$

où σ est la fonction sigmoïde.

— $\theta \in (0, 1)$: seuil de décision appliqué au score φ. Un span est étiqueté de type t si $\varphi(i, j, t) \geq \theta$, typiquement $\theta = 0.5$.

— $s_p = (i_p,\, j_p)$ et $s_g = (i_g,\, j_g)$: deux spans (prévision et vérité-terrain).

  — *Exact match* : $s_p = s_g$ *et* même label.

  — *Partial match (Jaccard)* : on calcule

$$\text{Jaccard}(s_p, s_g) = \frac{|\,\{i_p, \ldots, j_{p-1}\} \cap \{i_g, \ldots, j_{g-1}\}|}{|\,\{i_p, \ldots, j_{p-1}\} \cup \{i_g, \ldots, j_{g-1}\}|},$$

  et on considère le match validé si $\text{Jaccard}(s_p, s_g) \geq 0.5$ *et* labels identiques.

— Statuts de correspondance pour chaque prédiction vs. annotation : Avant d'aborder les métriques individuelles, il est essentiel de comprendre les quatre résultats fondamentaux d'un modèle de classification binaire, souvent représentés dans une matrice de confusion :

  — **Vrai Positif (TP)** : Entités reconnues par la NER et qui correspondent à la vérité terrain .

  — **Vrai Négatif (TN)** : Entités qui prédisent correctement la classe négative.

  — **Faux Positif (FP) / Erreur de Type I** : Entités reconnues par la NER mais qui ne correspondent pas à la vérité terrain.

  — **Faux Négatif (FN) / Erreur de Type II** : Entités annotées dans la vérité terrain qui ne sont pas reconnues par la NER.

— Métriques utilisée :

  — **Précision (Positive Predictive Value)** : Formule : Précision $= \frac{\text{TP}}{\text{TP+FP}}$ Mesure la proportion d'instances correctement prédites comme positives parmi toutes les instances prédites comme positives. Cruciale lorsque le coût d'un Faux Positif est élevé.

  — **Rappel (Sensibilité, True Positive Rate)** : Formule : Rappel $= \frac{\text{TP}}{\text{TP+FN}}$ Mesure la proportion d'instances correctement prédites comme positives parmi toutes les instances réellement positives. Cruciale lorsque le coût d'un Faux Négatif est élevé.

  — **Score F1** : Formule : Score F1 $= 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision+Rappel}}$ Moyenne harmonique de la précision et du rappel, fournissant une métrique unique qui équilibre les

deux.Donc il est utile pour les jeux de données déséquilibrés.

— **Specificity** : $\frac{TN}{TP+FN}$. : Elle mesure la capacité du modèle à identifier correctement les exemples appartenant à la classe négative. Contrairement à la sensibilité (ou rappel), qui évalue l'aptitude à détecter les événements de la classe positive, la spécificité reflète la rigueur avec laquelle le modèle évite de faussement attribuer un exemple à la classe positive.[7]

3 Étude bibliographiques

En littérature , les premières méthodes en été mise en place à base de règles en trouve divers approches qui ont été développées pour améliorer la précision et la robustesse des systèmes de NER. Ces approches peuvent être regroupées en quatre grandes catégories, que je détaille ci-dessous :

3.1 Approches basées sur des règles et des dictionnaires :

Les premières approches de la NER reposaient sur des règles linguistiques écrites manuelles et des dictionnaires spécialisés. Ces méthodes utilisaient des expressions régulières et des patrons syntaxiques qui sont un structures représent des schémas récurrents du langage utilisés pour identifier des relations lexicales dans les corpus textuels pour identifier des entités telles que les noms de personnes, les organisations ou les lieux. Bien que précises dans des contextes spécifiques, ces approches étaient limitées en termes de couverture et nécessitaient une maintenance constante pour s'adapter aux évolutions linguistiques.

Exemple : Considérons la phrase suivante :

"Emmanuel Macron est le président de la République française."

Une approche basée sur des règles pourrait utiliser une expression régulière comme :

```
[A-Z][a-z]+ [A-Z][a-z]+
```

pour détecter les noms propres composés de deux mots commençant par une majuscule, et ainsi identifier **Emmanuel Macron** comme une entité de type `PERSON`.[6]

3.2 Approches basées sur les modèles statistiques :

Pour dépasser les limites des règles écrites à la main, la communauté a exploré dès les années 2000 des approches statistiques apprenant à partir de corpus annotés. Les premiers travaux ont utilisé des Modèles de Markov Cachés (HMM) pour modéliser la séquence d'étiquettes comme un processus génératif séquentiel. Par exemple, Bikel (1997) a appliqué des HMM au NER pour l'anglais, et les Support Vector Machines

(SVM) ont également été testés au début des années 2000 . Toutefois, les HMM souffrent de limitations intrinsèques : en tant que modèles génératifs, ils calculent une probabilité jointe des mots et des étiquettes, ce qui complique l'intégration de multiples caractéristiques et de dépendances de long terme . Les SVM, de leur côté, traitent le NER comme un problème de classification binaire pour chaque mot ou chaque segment, nécessitant souvent d'entraîner un classifieur par type d'entité et par position, ce qui n'est pas trivial à combiner en sortie séquentielle . L'introduction des Conditional Random Fields (CRF) par Lafferty et al. (2001) a marqué une avancée significative. Les CRF sont des modèles discriminatifs conditionnels qui modélisent directement la probabilité de la séquence d'étiquettes étant donné la séquence de mots . Contrairement aux HMM, les CRF permettent d'intégrer un grand nombre de features dépendant du contexte sans hypothèse d'indépendance forte, tout en tenant compte des dépendances entre étiquettes voisines (grâce à la modélisation de la séquence entière) En pratique, les CRF se sont rapidement imposés comme la méthode de référence pour le NER durant les années 2000, obtenant de bons résultats sur divers corpus (par ex. CoNLL-2003 en anglais, ou des corpus spécifiques comme ceux des challenges BioNLP pour le biomédical)

Un système CRF typique pour le NER utilise de nombreuses caractéristiques conçues manuellement : formes des mots (majuscule, suffixe, préfixe), étiquettes grammaticales, appartenance à des listes d'entités, contexte lexical voisin, etc.

Cette ingénierie de features était essentielle pour obtenir de bonnes performances avec les modèles statistiques traditionnels, mais elle demandait une expertise humaine importante et n'était pas facilement généralisable à de nouveaux domaines. Bilan des approches classiques : Les méthodes à base de règles offraient précision et contrôle dans des cas d'usage restreints, mais manquaient de robustesse dès que le domaine changeait. Les modèles supervisés tels que HMM, CRF ou SVM ont apporté un cadre statistique rigoureux et une meilleure généralisation en apprenant à partir de données annotées. Cependant, ils exigeaient des features élaborées manuellement et restaient limités face à des phénomènes complexes (dépendances à longue portée, ambiguïtés subtiles). L'évolution suivante, marquée par l'essor de l'apprentissage automatique deep learning, allait précisément s'attacher à lever ces limites.

L'introduction de modèles statistiques a marqué une avancée significative en NER via les modèles de Markov cachés (Hidden Markov Model - HMM) qui est une méthode statistique permet de modéliser des séquences d'observations en supposant que ces observations sont générées par un processus de Markov caché.En effet,les HMM ont été parmi les premiers à être utilisés, modélisant les séquences de mots et leurs étiquettes correspondantes. Cependant, les HMM supposent une indépendance conditionnelle entre les observations, ce qui peut être restrictif. Pour surmonter cette limitation, les champs aléatoires conditionnels (conditional random fields ou CRFs) ont été introduits,

permettant de modéliser les dépendances entre les étiquettes sans faire d'hypothèses fortes sur les observations. Les CRF se sont révélés particulièrement efficaces pour la reconnaissance d'entités nommées, en particulier lorsqu'ils sont enrichis par des caractéristiques linguistiques pertinentes. [9]

3.3 Approches basées sur les réseaux de neurones récurrents (RNN) et BiLSTM-CRF :

Avec l'avènement de l'apprentissage profond et les réseaux de neurones récurrents (RNN), en particulier les réseaux LSTM (Long Short-Term Memory), ont été appliqués à la NER.En fait les modèles BiLSTM (Bidirectional LSTM) capturent le contexte des mots à la fois dans les directions avant et arrière, améliorant ainsi la reconnaissance des entités. Pour tirer parti des avantages des CRF et des BiLSTM, des architectures hybrides BiLSTM-CRF ont été développées. Ces modèles combinent la capacité des BiLSTM à capturer le contexte avec la capacité des CRF à modéliser les dépendances entre les étiquettes, offrant ainsi des performances de pointe en NER. [8]

3.4 Modèles Transformer pré-entraînés de type Transformer :

L'année 2018 a marqué un tournant avec l'introduction des **Transformers** pré-entraînés, en particulier le modèle **BERT** (Bidirectional Encoder Representations from Transformers) de Devlin et al. . BERT est un encodeur transformeur entraîné de façon auto-supervisée sur un très grand corpus (Wikipedia, livres) via des tâches de complétion de texte et de détection de phrases suivantes. Fine-tuné ensuite sur une tâche de NER supervisée, BERT a atteint des performances records, surpassant les BiLSTM-CRF de précédente génération. L'avantage majeur de BERT est d'offrir des embeddings de tokens profondément contextuels dès l'entrée du classifieur NER, rendant souvent superflue l'utilisation d'une couche CRF . En pratique, de nombreux systèmes se contentent d'ajouter une simple couche de classification sur les embeddings BERT pour chaque token, tout en obtenant d'excellents scores de F1. La « révolution BERT » a rapidement été déclinée dans de nombreuses langues et variantes spécialisées. Par exemple, pour le français, le modèle **CamemBERT** a été introduit en 2020 en entraînant l'architecture de BERT sur un large corpus français (textes issus de *Common Crawl* et Wikipédia). CamemBERT a permis d'améliorer nettement les performances des systèmes de NER en français par rapport aux modèles multilingues ou aux modèles entraînés sur l'anglais. De même, des modèles comme **XLM-R** (Conneau et al., 2019) ont offert des vecteurs multilingues robustes, donnant des résultats satisfaisants pour les langues peu dotées en données annotées. L'usage des Transformers a également favorisé l'émergence de nouvelles formulations de la tâche de NER (au-delà du simple étiquetage séquentiel) grâce à leur flexibilité. Par exemple, certains travaux ont abordé la

reconnaissance d'entités comme un problème de **question-réponse** en langage naturel , où l'on pose une question du type « Quel est le [TYPE D'ENTITÉ] dans le texte ? » et le modèle, basé sur BERT, extrait le segment de réponse correspondant dans le texte. D'autres approches ont formulé le NER comme une tâche de **génération libre de texte**, le modèle devant produire directement la liste des entités sous forme textuelle. Ces nouvelles perspectives ont été rendues possibles par la capacité des modèles Transformers à comprendre des instructions en langage naturel et à produire du texte, ouvrant la voie à l'utilisation des très grands modèles de langage. [13]

3.5 NER et grands modèles de langage (LLMs, *prompting*)

L'apparition des **grands modèles de langage** (LLM) tels que GPT-3 et ses successeurs (GPT-3.5, GPT-4, etc.) a suscité un intérêt pour des approches de NER sans entraînement spécifique, par simple *prompting*. En effet, ces modèles, entraînés sur d'énormes volumes de texte, possèdent une connaissance implicite des entités et peuvent, moyennant une consigne appropriée en langage naturel, extraire des entités mentionnées dans un texte donné. Par exemple, on peut fournir à ChatGPT une consigne du type : « Extrais toutes les personnes, organisations et lieux mentionnés dans le texte suivant... », et obtenir une réponse relativement correcte. Toutefois, des études récentes ont montré que, malgré des résultats prometteurs, le *prompting* direct de LLM génériques ne rivalise pas encore avec des modèles entraînés spécifiquement pour le NER [1]. Les LLM ont tendance à manquer de cohérence dans les réponses, peuvent halluciner des entités non présentes, et leur performance dépend fortement de la formulation de la requête (prompt). De plus, l'exploitation de ces modèles géants pose des problèmes pratiques de coût et de latence : interroger une API de LLM pour chaque document à traiter peut s'avérer prohibitif en production, sans parler des contraintes de confidentialité lorsque les données ne peuvent quitter une entreprise.

Pour combler cet écart, plusieurs travaux ont exploré l'adaptation ou la compression de LLM pour le NER. Certains ont cherché à **affiner (fine-tune)** directement de grands modèles comme T5 ou GPT-3 sur des corpus annotés en NER [2], mais cette approche est lourde en calcul et souvent irréalisable sans infrastructure spécialisée. D'autres ont proposé de **distiller** les connaissances d'un LLM vers un modèle plus petit dédié au NER. Par exemple,Pr.Zhou [14] introduisent **UniversalNER**, qui transfère sélectivement les capacités de ChatGPT vers un modèle plus compact afin de gérer la détection d'entités de tout type (*open NER*). Ces efforts témoignent d'une convergence des paradigmes : utiliser les LLM comme sources de savoir ou d'annotation pour améliorer des modèles ciblés, tout en conservant l'efficacité de ces derniers.

3.6 GLiNER et nouvelles approches (InstructNER, SpanNER)

3.6.1 GLiNER : un modèle généraliste pour le NER

Face aux limites des LLMs en production, **GLiNER** (*Generalist Lightweight NER*) a été proposé par Zaratiana et al. en 2023 [12] et présenté à NAACL 2024 et développé par F.initiatives et la Laboratoire Informatique de Paris Nord :

(a) F.initiatives

(b) Laboratoire Informatique de Paris Nord

Il présent une solution intermédiaire alliant la flexibilité des LLM et l'efficacité de modèles plus petits. GLiNER est un modèle de NER capable d'identifier des entités de n'importe quel type, sans être limité à une liste fixe de catégories prédéfinies. Pour ce faire, il utilise un **encodeur bidirectionnel de type BERT** et prend en entrée non seulement le texte à analyser, mais aussi la liste des types d'entités à extraire (spécifiée sous forme de mots ou de courtes descriptions fournies en entrée). Concrètement, GLiNER concatène les types d'entités requis avec le texte, en les séparant par un token spécial (par exemple [ENT]), puis encode l'ensemble. Il en résulte des vecteurs de représentation aussi bien pour chaque token du texte que pour chaque type d'entité. Le modèle calcule alors, pour chaque segment de texte possible (suite de tokens consécutifs) et pour chaque type d'entité, un score de correspondance indiquant la probabilité que ce segment soit une entité de ce type [12]. Tous ces calculs étant effectués en parallèle dans l'espace vectoriel, GLiNER peut extraire simultanément plusieurs entités de types différents, contrastant ainsi avec la génération token par token des LLM qui est plus lente [12]

En termes de performances, GLiNER se montre très prometteur. En évaluation *zero-shot* (sans aucun ajustement sur le domaine ou le jeu de données cible), il surpasse non seulement les modèles de NER classiques entraînés sur des catégories fixes, mais également ChatGPT et d'autres LLM lorsqu'ils sont sollicités pour extraire des entités sans entraînement spécifique [12]. Cette supériorité a été démontrée sur divers jeux de données de référence en NER. De plus, GLiNER, de par sa taille modeste (environ 300 millions de paramètres pour la version base, soit quelques gigaoctets), peut être déployé sur des machines raisonnablement puissantes y compris sans GPU, ce qui est hors de portée des LLM contenant des dizaines de milliards de paramètres. Notons que GLiNER supporte le multilingue : le modèle publié [12] a été entraîné principalement sur de l'anglais, mais il montre de bonnes performances en français, allemand, espagnol, etc., grâce à l'inclusion de données variées lors de son entraînement. Son code et modèle

pré-entrainé sont disponibles librement [1], ce qui facilite son adoption et son extension.

3.6.2 InstructNER : alignement multi-modal pour le NER

Parallèlement à GLiNER, d'autres approches récentes cherchent à repousser les limites du NER traditionnel en exploitant de nouvelles sources d'information. **InstructNER** [11] est l'une d'elles : il s'agit d'un cadre qui transforme la tâche de NER classique (sur du texte seul) en une tâche **multimodale** en combinant texte et image. L'idée part du constat que certaines ambiguïtés dans le texte pourraient être levées si l'on disposait d'indices visuels. InstructNER génère donc, pour chaque phrase à analyser, une image synthétique censée représenter visuellement le contenu de la phrase (à l'aide d'un modèle de diffusion du type Stable Diffusion pré-entraîné). Ensuite, le système sélectionne l'image synthétique la plus pertinente en la comparant à de véritables images trouvées sur Internet liées aux entités du texte, grâce à une mesure de similarité visuelle [11]. Enfin, l'architecture entraîne un modèle de NER multimodal qui prend en entrée à la fois le texte et l'image choisie, en utilisant un mécanisme d'attention croisée pour aligner les deux modalités. Une composante d'*attention guidée* est ajoutée pour encourager le modèle à se focaliser sur les zones de l'image pertinentes pour la reconnaissance des entités. Les expériences menées par Wang et al. [11] montrent que InstructNER améliore les performances par rapport aux modèles purement textuels sur des jeux de données de NER multimédia, avec un gain de F1 entre +1,4% et +2,3%. Même face à d'autres modèles multimodaux complets (c'est-à-dire entraînés avec de vraies images en entrée), InstructNER reste compétitif tout en n'ayant besoin que d'images synthétiques. Cette approche illustre une direction de recherche visant à enrichir le contexte de NER au-delà du seul texte brut. Le code source et un jeu de données d'images synthétiques générées pour ces expériences ont d'ailleurs été mis à disposition par les auteurs [11] .

3.6.3 SpanNER : NER par classification de segments

Une autre évolution notable est le retour au cœur du texte via les **approches par segments** (*span-based*) plutôt que par étiquetage token par token. **SpanNER** de Fu et al. est représentatif de cette catégorie. Au lieu d'attribuer à chaque mot une étiquette, SpanNER considère toutes les sous-séquences possibles de mots (appelées segments ou *spans*) dans une phrase et prédit lesquelles correspondent à des entités nommées et de quel type. Pour ce faire, le modèle génère d'abord des représentations vectorielles pour chaque segment candidat (par exemple en combinant les embeddings du premier et du dernier mot du segment, ainsi que peut-être d'autres indicateurs comme la longueur). Ensuite, un classifieur binaire est entraîné pour chaque type d'entité afin de déterminer si un segment donné est une entité de ce type. Cette approche exploite souvent une

1. https://github.com/urchade/GLiNER

architecture de type Transformer ou BiLSTM pour encoder la phrase, puis applique un réseau de neurones feed-forward sur les vecteurs de segments. SpanNER présente plusieurs intérêts : il peut détecter naturellement des entités **imbriquées** (une entité plus longue en contenant une plus courte, ce qui arrive par exemple avec des titres d'œuvres contenues dans des entités de localisation, etc.) car chaque segment est évalué indépendamment ; de plus, il n'impose pas de structurer la sortie sous forme d'une séquence d'étiquettes de même longueur que le texte, ce qui permet plus de flexibilité. Les résultats reportés par Pr.Fu montrent que SpanNER atteint des performances comparables aux meilleurs modèles séquentiels sur le NER en anglais, tout en offrant ces avantages structurels. Depuis, d'autres travaux ont combiné l'approche par segments avec des modèles neuronaux plus récents (par exemple en utilisant BERT pour encoder les segments) et ont proposé des améliorations pour réduire le très grand nombre de segments candidats (car une phrase de n mots possède $O(n^2)$ segments possibles). Néanmoins, la formulation par segments s'est établie comme une alternative crédible et puissante, et GLiNER lui-même peut être vu comme une version optimisée de cette idée, où les segments sont évalués simultanément via des produits scalaires dans l'espace latent.

3.6.4 Tableau comparatifs des différent approches

Approche	Points forts	Limites
Règles et dictionnaires	- Grande précision sur des schémas bien définis - Contrôle total par l'expert humain	- Pas de couverture hors contexte connu - Faible adaptabilité aux nouveaux domaines
Modèles statistiques (HMM, CRF, SVM)	- Bonne généralisation sur corpus annoté - Capacité à modéliser le contexte (CRF) - Performances solides sur des jeux de données standards	- Dépendance à des features définies manuellement - Moins efficaces sur des phénomènes complexes (dépendances longues) - Nécessitent beaucoup de données annotées
Réseaux neuronaux (BiLSTM, BiLSTM-CRF)	- Capturent automatiquement le contexte - Bonnes performances en NER séquentiel	- Nécessitent des ressources de calcul importantes - Limités sur entités imbriquées - Dépendance au volume de données annotées
Transformers préentraînés (BERT, CamemBERT)	- Embeddings contextuels puissants - Excellentes performances - Peu ou pas d'ingénierie de features nécessaire	- Modèles volumineux (mémoire, calcul) - Limités aux types d'entités appris pendant l'entraînement - Moins flexibles pour de nouveaux schémas d'entités
LLMs (GPT, Llama) avec prompting	- Flexibilité extrême (zero-shot) - Capables de traiter des consignes en langage naturel - Potentiel sur langues peu dotées	- Coût très élevé à l'inférence - Risques d'hallucinations
Span-based NER (SpanNER)	- Capable de détecter des entités imbriquées - Plus flexible que l'étiquetage séquentiel - Bonnes performances comparables aux CRF/Transformers	- Coût calculatoire élevé pour longs textes - Nécessite des techniques pour réduire l'espace de recherche
GLiNER	- Détecte dynamiquement n'importe quel type d'entité - Léger et rapide (vs LLMs) - Fonctionne en zero-shot sur nouveaux domaines - Multilingue performant	- Moins puissant qu'un grand LLM sur certains contextes très spécifiques - Dépend encore d'un encodeur Transformer donc coûte plus cher qu'un simple CRF

3.7 Conclusion

Les travaux récents illustrés par GLiNER, InstructNER, SpanNER et d'autres montrent que la recherche en NER est plus active que jamais.

On observe en particulier une tendance à concilier la *généralité* (être capable de détecter tout type d'entité selon les besoins, y compris des types définis dynamiquement) et la *spécialisation* (bénéficier de modèles compacts, rapides et entraînés pour une tâche précise). GLiNER vise précisément cet équilibre en s'inspirant des capacités ouvertes des LLM tout en restant léger et déployable.

En conclusion, la tâche de reconnaissance d'entités nommées continue d'évoluer au gré des innovations en apprentissage automatique. Si les défis persistent (adaptation à de nouveaux domaines, gestion des entités rares ou à définition floue, optimisation des ressources), les avancées récentes laissent présager des systèmes de plus en plus robustes, polyvalents et intégrés à des applications complexes allant au-delà de la simple extraction d'entités, pour structurer la connaissance contenue dans les textes.

Au regard de cette évolution et de l'analyse comparative menée, mon choix s'est porté sur **GLiNER** pour la suite de ce travail. En effet, GLiNER présente plusieurs atouts déterminants :

- Il est capable d'identifier des entités de *n'importe quel type*, y compris des catégories non vues durant l'entraînement, grâce à sa conception orientée zéro-shot.
- Il combine la puissance des modèles Transformers avec une architecture optimisée, offrant un bon compromis entre **précision**, **flexibilité** et **efficacité computationnelle**.
- Il est **open-source** et plus léger que les grands modèles de langage (LLMs), ce qui permet un déploiement plus réaliste dans des environnements à ressources limitées.
- Il dispose d'une **capacité multilingue** appréciable, atout essentiel dans un contexte de diversité linguistique.

Ces caractéristiques font de GLiNER une solution particulièrement adaptée à mes objectifs, en me permettant d'explorer la reconnaissance d'entités nommées dans des contextes variés, tout en maîtrisant les coûts en termes de ressources et de complexité de déploiement. GLiNER sera donc le cœur de l'approche proposée dans la suite de ce rapport .

4 Décomposition fonctionnelle de GLiNER

Dans cette partie je vais commence par un exemple concret puis je voulais décrire en détaille les composantes internes de GLiNER .

4.1 Exemple concret : Entrée / sortie

Le modèle GLiNER prend en entrée une séquence combinant les types d'entités et le texte cible. Par exemple :

$$\underbrace{\texttt{[ENT] person [ENT] Disorder [SEP]}}_{\text{zone « types des entités »}} + \underbrace{\textit{"Patients with idiopathic generalized epilepsies "}}_{\text{zone « texte »}}$$

Texte extrait	**Type prédit**	**Span (start, end)**
Patients	person	$(0, 8)$
idiopathic generalized epilepsies	Disorder	$(14, 47)$

Table 2 – Sortie typique retournée par GLiNER.

4.2 Composants internes

L'architecture de GLiNER repose sur l'interaction entre trois composants majeurs : un encodeur bidirectionnel de type bert, un mécanisme de représentation des entités (labels), et un module de représentation des spans textuels. Ces composants sont conçus pour projeter de manière conjointe les types d'entités et les fragments de texte dans un espace latent commun, facilitant une correspondance flexible entre eux comme explique dans la graphe ci dessous : .

— **Architecture de GLiNER :**

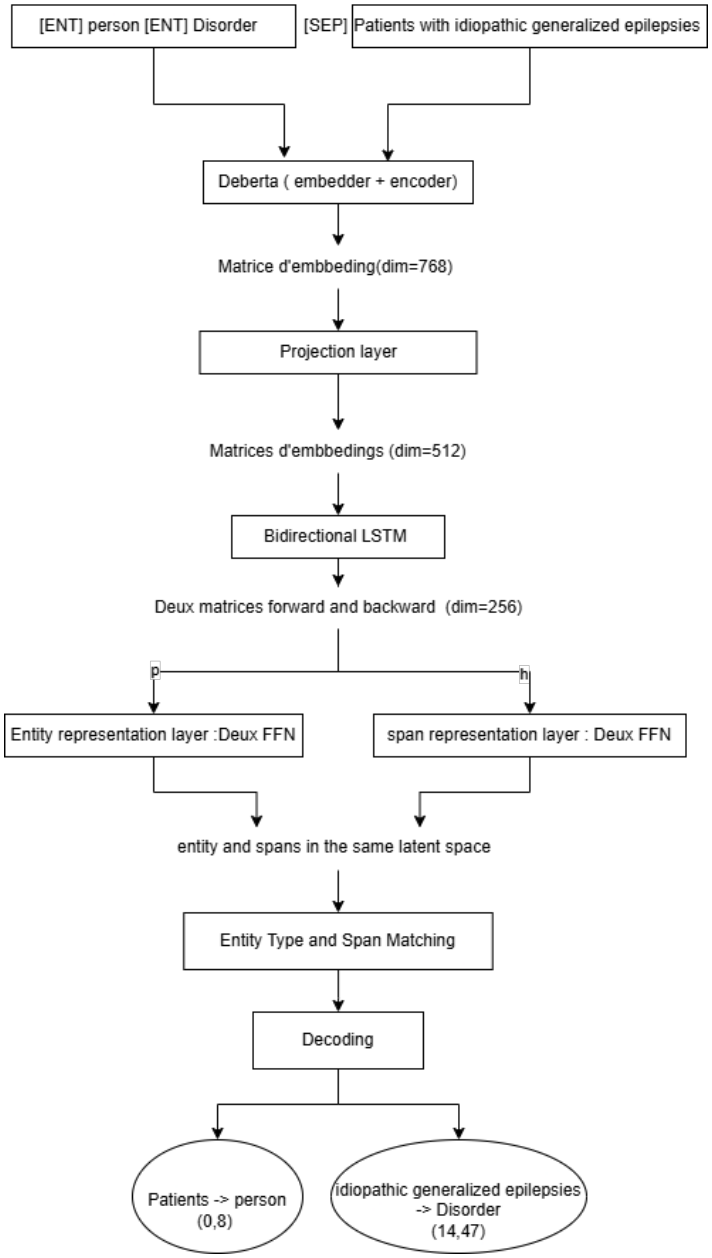


Figure 2 – Architecture détaillée du modèle GLiNER.

Soit $\mathbf{p} = \{\mathbf{p}_i\}_0^{M-1} \in \mathbb{R}^{M \times D}$ la sortie de l'encodeur pour chaque type d'entité, correspondant à toutes les représentations des tokens [ENT]. De même, $\mathbf{h} = \{\mathbf{h}_i\}_0^{N-1} \in \mathbb{R}^{N \times D}$ désigne la représentation de chaque mot dans le texte d'entrée.

En effet cette architucture détaille les étapes interne dans GLiNER qui lui spécifier du l'input jusqu'à l'ouput du model , donc par la suite je vais détaille chaque composant du model , donc la premier partie est :

— **1.Tokenisation BPE (Byte-Pair Encoding) utilisée dans DeBERTa**

La tokenisation constitue une étape fondamentale dans tout pipeline de traitement automatique du langage naturel. Dans le cas de DeBERTa, le modèle repose sur une version spécifique de la tokenisation appelée **Byte-Pair Encoding**

(BPE), une méthode efficace et largement adoptée dans les modèles modernes de type Transformer.

Le BPE, à l'origine conçu comme un algorithme de compression de texte, a été adapté pour générer dynamiquement des unités de sous-mots (*subwords*) à partir d'un corpus d'entraînement. Plutôt que de travailler avec un vocabulaire figé de mots entiers, cette méthode permet de découper les mots rares en unités plus petites et plus fréquentes, ce qui améliore la couverture du vocabulaire tout en gardant une taille de vocabulaire raisonnable. Chaque mot est initialement représenté comme une suite de caractères, puis l'algorithme identifie les paires de caractères les plus fréquentes et les fusionne. Cette opération est répétée de manière itérative jusqu'à atteindre la taille de vocabulaire souhaitée. Par exemple, si les mots "hug", "pug" et "hugs" apparaissent fréquemment dans le corpus, l'algorithme apprendra à fusionner les caractères "u" et "g" en "ug", puis "h" et "ug" en "hug", etc.

Ce procédé est à la base de la construction du vocabulaire utilisé pour encoder les textes. Grâce à ces avantages , d'une part, il permet de représenter n'importe quel mot, même s'il n'a jamais été vu pendant l'entraînement, en le décomposant en sous-mots connus ; d'autre part, il réduit considérablement le nombre de token inconnus (`[UNK]`), tout en préservant l'information contextuelle utile pour des tâches comme la reconnaissance d'entités nommées.

Certains modèles comme **RoBERTa** vont encore plus loin dans le traitement des entrées en appliquant la tokenisation BPE non pas sur des caractères Unicode, mais directement sur les **bytes**. Cette variante, appelée *byte-level BPE*, permet de garantir que tous les caractères – y compris des emojis, symboles rares ou lettres non latines – soient toujours représentables dans le vocabulaire du modèle. Grâce à cette approche, aucun token inconnu (`[UNK]`) n'est généré, ce qui améliore considérablement la robustesse du modèle face à des entrées inhabituelles ou bruitées.

Le modèle **DeBERTa**, tout en conservant cette logique BPE classique, introduit deux innovations majeures qui le distinguent des autres modèles Transformer classiques comme BERT :

A. Partage des embeddings (*embedding sharing*)

Cette technique consiste à utiliser le même espace vectoriel pour encoder les sous-mots à l'entrée du modèle et pour les décoder ou les classifier en sortie. En d'autres termes, les vecteurs d'embedding sont partagés entre l'embedding d'entrée et la tête de sortie. Cette approche réduit drastiquement le nombre de paramètres à apprendre, rendant le modèle plus léger et plus stable. Elle favorise également une meilleure cohérence entre les représentations internes et

les sorties générées, ce qui peut se traduire par une amélioration des performances, notamment en apprentissage supervisé.

B. Attention désentrelacée (*disentangled attention*)
Dans les Transformers traditionnels, les vecteurs de chaque token combinent à la fois l'information lexicale (le mot lui-même) et sa position dans la séquence. Ces deux dimensions sont fusionnées dans un unique vecteur, ce qui limite parfois la capacité du modèle à les traiter de façon distincte. DeBERTa propose de désentrelacer ces deux informations : le contenu lexical et la position sont encodés séparément à l'aide de deux vecteurs distincts, puis combinés au moment de l'attention. Ce mécanisme permet au modèle de raisonner indépendamment sur «quoi» (le contenu) et «où» (la position), améliorant ainsi sa capacité à capturer les relations sémantiques et syntaxiques complexes au sein du texte. [3]

En résumé, la tokenisation BPE utilisée dans DeBERTa, combinée à ces innovations structurelles, joue un rôle fondamental dans la robustesse et la flexibilité du modèle. Elle permet de découper les mots en unités fréquentes et réutilisables, tout en assurant une compatibilité parfaite avec le système d'alignement des entités basé sur les spans. Ces caractéristiques en font un choix particulièrement pertinent pour des tâches comme la reconnaissance d'entités nommées (NER) sur des corpus spécialisés ou techniques.

— **2. Représentation des types d'entité et du text (prompts)**
Pour rendre l'extraction d'entités *ouverte* (*open-type*), chaque type d'entité à détecter est exprimé sous forme d'un prompt textuel (par exemple `"disease"`, `"chemical"`), précédé d'un token spécial `[ENT]`. Ces prompts sont concaténés au début de la séquence d'entrée, puis séparés du texte source par un token `[SEP]` :

```
[ENT] disease [ENT] chemical [SEP] Le patient présente une ...
```

L'encodeur traite l'ensemble de la séquence et produit une représentation contextualisée pour chaque token.

— **3.Encodeur bidirectionnel (BERT/DeBERTa) :**
Le cœur de GLiNER repose sur un *encodeur bidirectionnel* de type BERT/-DeBERTa, qui permet de modéliser simultanément le contexte gauche et le contexte droit de chaque token. Contrairement aux architectures auto-regressives, cet encodeur traite l'intégralité de la séquence en un seul passage, générant pour chaque position un vecteur contextuel enrichi par l'attention sur tous les autres tokens. Cette capacité à capturer à la fois les informations antécédentes et suivantes est essentiel pour différencier des entités homographes (Il désigne deux mots ou plus ayant une orthographe identique, mais un sens différent) en

fonction de leur environnement linguistique.

Après l'encodage de la prompt , deux sous-ensembles sont extraits :

— $\{p_i\}$: représentations des types d'entité (tokens précédant le `[SEP]`),

— $\{h_j\}$: représentations des tokens du texte (tokens suivant le `[SEP]`).

Les représentations $\{p_i\}$ sont ensuite raffinées par un réseau feed-forward pour produire les vecteurs finaux $\{q_t\}$, correspondant à chaque type d'entité.

— **3.Affinage des représentations des types d'entité :**

Après l'encodage de la séquence complète (types d'entités + texte), les vecteurs $\{p_i\}$ et $\{h_j\}$ capturent le contexte des types dans la séquence complète, incluant leurs interactions mutuelles et avec le texte.

Cependant, pour assurer une correspondance précise avec les représentations des spans textuels $\{S_{ij}\}$, ces vecteurs $\{p_i\}$ sont projetés dans un espace latent homogène via un réseau de neurones feed-forward à deux couches. Ce réseau a pour but d'aligner la dimension des types avec celle des représentations des spans.

Concrètement, chaque vecteur p_i (représentation initiale du type t_i) est transformé selon :

$$z_t = W_t^{(1)} p_i + b_t^{(1)}$$
$$a_t = \text{GELU}(z_t)$$
$$q_t = W_t^{(2)} a_t + b_t^{(2)}$$

où :

— $p_i \in \mathbb{R}^D$ est l'entrée initiale (embedding du type),

— $q_t \in \mathbb{R}^{D'}$ est la sortie raffinée, utilisée pour le matching avec les spans,

— $W_t^{(1)}$ et $W_t^{(2)}$ sont des matrices de poids apprises, spécifiques à cette projection,

— GELU est une fonction d'activation non linéaire douce, décrite par :

$$\text{GELU}(x) = x \cdot \frac{1}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

Ce processus de projection permet d'obtenir une représentation vectorielle q_t pour chaque type d'entité, apte à être directement comparée à un vecteur de span S_{ij} lors de la phase de prédiction.

La fonction d'erreur utilisée dans la $\text{GELU}(x)$ est $\text{erf}(x)$ est définie par l'intégrale suivante :

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

Elle permet d'approximer la probabilité qu'une variable aléatoire suivant une loi normale centrée réduite soit inférieure à x.

— **4. Représentation des spans textuels**

Parallèlement, GLiNER génère tous les sous-spans (i, j) du texte d'entrée (avec $j - i \leq K = 12$), considérés comme candidats à une détection d'entité. Pour chaque span, un vecteur S_{ij} est calculé via les étapes suivantes :

1. **Extraction des contextes** : on récupère h_i et h_j, vecteurs aux positions de début et de fin du span.
2. **Concaténation** : on forme tous les combinainsant possible des vecteurs $[h_i; h_j] \in \mathbb{R}^{2D}$.
3. **Projection non linéaire** :On applique la projection de $[h_i; h_j]$ par double FFN :

$$z = W_1 [h_i; h_j] + b_1,$$
$$a = \text{GELU}(z),$$
$$S_{ij} \;=\; \text{FFN}\Big(h_i \otimes h_j\Big) = W_2\, a + b_2$$

S_{ij} capture la sémantique du fragment de texte considéré, sous forme vectorielle et reporesnete en meme espace latente que l

— **5. Mécanisme de correspondance span–type :**

Enfin, pour chaque couple (span, type) on calcule un score de compatibilité :

$$\varphi(i, j, t) \;=\; \sigma\Big(S_{ij}^{\top} q_t\Big)$$

où q_t (représentation de l'entité) est l'embedding du type t et σ la fonction sigmoïde. Ce score, interprété comme la probabilité que le span (i, j) appartienne au type t, est comparé à un seuil (typiquement 0,5) pour décider de l'étiquetage. Cette stratégie de *matching* binaire permet d'assigner plusieurs types à un même span et de conserver une complexité en $\mathcal{O}(nK)$.

5 Méthodologie proposée pour enrichir GLiNER

5.1 Motivation de l'évolution du modèle

Le modèle GLiNER repose sur un paradigme *open-type*, dans lequel les entités à reconnaître sont spécifiées dynamiquement sous forme de libellés textuels, comme *"disease"* ou *"chemical"*. Toutefois, cette formulation se limite généralement à des noms d'entités courts, ce qui réduit la richesse sémantique accessible par le modèle lors de l'encodage des types.

Dans des contextes complexes (biomédecine, droit, etc.), une simple mention nominale peut ne suffit souvent pas à capturer les subtilités du type d'entité recherché. Pour surmonter cette limitation, nous proposons d'enrichir la représentation des entités via leur **définition textuelle complète**, permettant ainsi au modèle d'exploiter davantage d'information sémantique et contextuelle.

Nous présentons deux stratégies pour atteindre cette objectif :

— **(S1)** : **Fine-tuning de GLiNER avec des définitions d'entités comme libellés**.

— **(S2)** : **Changement du modèle d'embedding et encoder utilisé pour les libellés**.

5.2 Stratégie 1 – Fine-tuning avec des définitions comme libellés

5.2.1 Principe général

Dans cette première approche, nous conservons l'architecture originale de GLiNER, mais modifions la formulation des libellés fournis au modèle. Plus précisément, au lieu d'utiliser des noms simples de types d'entités (par exemple `"disease"`), nous injectons une **définition textuelle complète**, comme :

```
[ENT] Any deviation from the normal state of an organism: diseases,
symptoms, dysfunctions, organ abnormalities (excluding injuries
or poisoning). [SEP] Text...
```

Cette reformulation permet de fournir un contexte sémantique beaucoup plus riche pour chaque type d'entité à détecter. Le modèle est alors entraîné sur un corpus annoté où les types sont définis par des descriptions explicites plutôt que des simples mots-clés.

5.2.2 Définition et rôle du *fine-tuning*

Le **fine-tuning** est une méthode d'apprentissage supervisé visant à adapter un modèle pré-entraîné sur une large collection de données génériques à une tâche spécifique ou à un domaine particulier. Plutôt que de réentraîner un modèle depuis zéro, on conserve ses poids initiaux (issus du pré-entraînement) et on poursuit l'apprentissage sur un nouveau jeu de données plus ciblé. Cela permet de bénéficier à la fois de la robustesse des représentations acquises et de la spécialisation nécessaire pour de nouveaux cas d'usage.

Dans notre cas, nous avons utilisé le modèle GLiNER, un modèle de reconnaissance d'entités nommées (*Named Entity Recognition*) conçu pour fonctionner avec des types d'entités formulés sous forme de mots ou de courtes descriptions. Le fine-tuning ne vise pas à changer la tâche de base (la détection d'entités nommées dans un texte), mais à adapter le modèle à une nouvelle formulation des consignes : au lieu d'associer chaque entité à un mot-clé simple (ex. “disease”, “protein”), nous utilisons une définition complète et descriptive du type d'entité, souvent issue d'un vocabulaire biomédical contrôlé.

Cette modification introduit une plus grande richesse sémantique, mais également une plus grande complexité, car le modèle doit maintenant apprendre à faire correspondre des segments textuels à des consignes plus longues, parfois ambigües ou redondantes. Le fine-tuning permet donc :

— d'adapter le modèle à la structure linguistique et au contenu spécifique de notre corpus biomédical annoté ;
— d'améliorer la capacité du modèle à généraliser à des entités rarement vues ou nouvelles, en exploitant les similarités sémantiques dans les définitions ;
— d'apprendre à utiliser des descriptions longues et complexes comme vecteurs d'information lors du processus de détection.

Avantages par rapport à un entraînement *from scratch* Le fine-tuning présente de nombreux avantages par rapport à un entraînement complet à partir de zéro :

— IL nécessite moins de données annotées, car les connaissances générales du langage et des structures grammaticales sont déjà acquises ;
— IL est beaucoup plus rapide et moins coûteux computationnellement (notamment en temps et en mémoire GPU) ;
— IL limite le risque de surapprentissage, surtout lorsque le jeu de données spécifique est de taille modeste.

Choix et justification des hyperparamètres

Pour fine-tuner le modèle GLiNER, nous avons utilisé la classe `TrainingArguments` de la bibliothèque Hugging Face Transformers, en fixant manuellement plusieurs hyperparamètres critiques afin de garantir la stabilité de l'entraînement tout en tenant compte des contraintes de mémoire et des spécificités de notre tâche. Voici les paramètres retenus :

— **Nombre d'époques** (`num_train_epochs=20`) : nous avons choisi un nombre relativement élevé d'époques pour permettre un ajustement progressif, tout en activant le paramètre `load_best_model_at_end=True` afin de conserver le modèle avec la meilleure perte de validation.
— **Taille de batch par GPU** (`per_device_train_batch_size=2`) : limitée volontairement à 2 pour respecter les contraintes de mémoire sur GPU (16 Go), en combinaison avec `gradient_accumulation_steps=2` pour simuler un batch effectif de taille 4.
— **Précision mixte** (`fp16=True`) : cette option permet de réduire la consommation mémoire GPU et d'accélérer l'entraînement, en utilisant des calculs en virgule flottante 16 bits lorsque cela est possible.
— **Taux d'apprentissage principal** (`learning_rate=5e-6`) : une valeur faible a été choisie pour éviter toute déstabilisation brutale des poids déjà appris. Un

scheduler linéaire avec échauffement (`warmup_ratio=0.1`) a également été utilisé pour lisser la montée des gradients au début de l'apprentissage.

— **Taux d'apprentissage secondaire** (`others_lr=1e-5`) : ce taux est appliqué aux couches nouvellement initialisées (par exemple la tête de classification), qui doivent être entraînées plus rapidement que le backbone.

— **Régularisation** (`weight_decay=0.01`) : un terme de pénalisation L_2 a été introduit pour prévenir le surapprentissage, notamment sur les définitions longues qui pourraient entraîner un surajustement sémantique.

— **Stratégies d'évaluation et de sauvegarde** : nous avons choisi d'évaluer et de sauvegarder à chaque époque (`eval_strategy="epoch"`, `save_strategy="epoch"`) pour conserver les checkpoints clés. Le modèle avec la perte de validation la plus basse est automatiquement rechargé à la fin de l'entraînement.

— **Autres paramètres** : le nombre de travailleurs de chargement (`dataloader_num_workers=0`) a été fixé pour éviter les erreurs dans l'environnement Kaggle, et les logs ont été désactivés dans WandB (`report_to="none"`).

Ces réglages ont été testés empiriquement : nous avons observé qu'un taux d'apprentissage trop élevé ou un batch trop large entraînaient une divergence rapide du modèle ou des erreurs CUDA de dépassement mémoire. À l'inverse, les hyperparamètres retenus ont permis une convergence progressive avec une perte de validation stable dès les premières époques.

5.3 Stratégie 2 – Changement du modèle d'embedding et encoder utilisé pour les libellés

5.3.1 Objectif

Dans cette deuxième stratégie, nous conservons l'architecture principale de GLiNER, mais nous remplaçons uniquement le module dédié à l'encodage des libellés. Ce module joue un rôle fondamental : il transforme chaque nom d'entité cible en un vecteur dense. Ce vecteur est ensuite passé manuellement par un `pooler` puis par un LSTM bidirectionnel, afin d'obtenir une représentation finale qui partage la même dimension que les représentations des spans extraits du texte. Cette étape est essentielle pour permettre la comparaison entre ces deux représentation vectoriels.

5.3.2 Encodeur original vs encodeur proposé

Initialement, GLiNER utilise le modèle `BAAI/bge-small-en-v1.5`, qui produit des vecteurs de dimension 384. Afin d'améliorer la qualité des embeddings de libellés, nous remplaçons cet encodeur par `sentence-transformers/all-MiniLM-L6-v2`, un modèle de la famille transformers, connu pour son efficacité dans la similarité sémantique.

all-MiniLM génère des vecteurs de taille 384, ce qui est directement compatible avec l'architecture interne de GLiNER.

5.3.3 Approche technique mise en place

Pour intégrer MPNet comme encodeur de libellés, nous procédons comme suit :

1. **Encodage des libellés** : chaque libellé est encodé par MiniLM, ce qui produit un tenseur de dimension $[\text{num\_labels}, 384]$;
2. **Réduction et activation** : ce tenseur est ensuite traité par une couche de pooling `Linear + Tanh`, dont les poids sont directement repris du modèle GLiNER d'origine ;
3. **Modélisation séquentielle** : la sortie précédente alimente une LSTM bidirectionnelle, elle-même importée telle quelle du modèle initial.

Aucune couche de projection supplémentaire n'est ajoutée ni modifiée, contrairement à certaines variantes du modèle. Ainsi, les entités demeurent encodées dans l'espace latent exact attendu par le cœur du réseau, conformément à la conception originale.

5.3.4 Résultat de l'intégration

Grâce à cette substitution ciblée, nous obtenons un modèle capable de :
— exploite pleinement la richesse sémantique de MiniLM par rapport à la modéle de base ;
— préserve la compatibilité structurelle avec l'architecture GLiNER ;
Cette stratégie a permis d'augmenter la robustesse du modèle notamment dans les cas où les entités sont définies comme des définition de manière plus complexe ou plus nuancée .Mais pour le réussir en plein performance on dois fait un entraînement du modéle de nouveau pour que les poids soit synchronisée et pris en compte le changement au niveaux de l'encodeur .

5.4 Conclusion de la méthodologie

L'ajout de définitions enrichies, combiné à un encodeur sémantique adapté, permet à GLiNER de mieux capturer des entités. Cette amélioration est particulièrement pertinente en NER biomédical, où les types sont complexes et souvent ambigus. Les deux stratégies présentées peuvent être combinées pour optimiser à la fois l'architecture et les données d'entraînement.

6 Expérimentation

6.1 Présentation du jeu de données

BioASQ organise chaque année des tâches d'indexation sémantique et de question-réponse sur la littérature biomédicale ainsi que l'édition 2024 introduit notamment la sous-tâche BioNNE dédiée à la reconnaissance d'entités imbriquées .

Le jeu de données utilisé pour évaluer et affiner GLiNER rassemble des résumés PubMed issus du challenge BioASQ 2024. Ainsi que les abstracts ont été annotés dans le format (.ann) pour huit types d'entités biomédicales (DISO, ANATOMY, CHEM, FINDING, PHYS, LABPROC, INJURY_POISONING, DEVICE). Le sous-ensemble officiel « BioNNE » contient 54 abstracts d'entraînement, 50 de développement et 500 de test. Les textes proviennent de MEDLINE/PubMed (licence CC-BY 4.0) et sont maintenus par le National Library of Medicine .La dataset BioASQ fournit les fichiers texte (.txt) et annotation (.ann) . Nous avons fusionné ces paires, puis généré deux jeux JSON : un jeu *évaluation* (texte + annotation) et un jeu *fine-tuning* (texte tokenisé + annotation).

Le jeu de données sert à mesurer les performances du modèle **GLiNER** et de la fine-tuné sur des abstracts biomédicaux. J'ai effectuer un transformation des donnée consiste a fusionnée tous les fichier des texts et ses annotations afin d'obtenir deux fichier de format JSON l'un va être utilisée pour l'évaluation et l'autre pour le fine Tuning .

6.1.1 Détermination des synonyme :

Afin de mieux comprendre les nuances sémantiques associées à chaque type d'entité biomédicale de notre jeu de données (DISO, ANATOMY, CHEM, etc.) et d'identifier des synonymes pertinents, nous avons mis en place une méthodologie basée sur des modèles de langage pré-entraînés . Cette approche est particulièrement pertinente car elle s'appuie sur les même modèles employées par GLiNER.

Notre processus de détermination des synonymes repose sur l'utilisation combinée d'un Modèle de Langage Masqué (MLM) et d'un modèle d'embeddings sémantiques . Pour cette tâche, nous avons spécifiquement choisi le modèle 'bert-base-uncased' pour le MLM. Ce choix est délibéré : BERT est un transformeur bidirectionnel largement reconnu pour sa capacité à générer des représentations contextuelles profondes du langage, ce qui en fait un candidat idéal pour prédire des mots manquants en fonction de leur contexte. De plus, les architectures basées sur des transformeurs sont au cœur des modèles comme GLiNER, assurant une cohérence dans les principes sous-jacents de la représentation du langage. Pour la mesure de similarité sémantique, nous avons opté pour 'all-MiniLM-L6-v2', un Sentence Transformer efficace qui fournit des embeddings

de phrases de haute qualité, permettant une comparaison précise de la signification des mots.

L'algorithme de détermination des synonymes se déroule comme suit :

1. **Masquage du mot cible :** Pour chaque phrase d'analyse et le mot cible (par exemple, "chemical" dans "Today, I get the result of my blood tests that revealed the presence of a specific `[MASK]` biomarker."), le mot cible est remplacé par le token de masque spécial du tokenizer (‘[MASK]‘).
2. **Prédiction par le MLM :** Le modèle ‘bert-base-uncased‘ est ensuite utilisé pour prédire les 10 mots les plus probables pour remplacer le token masqué. Chaque prédiction est accompagnée d'un score de probabilité (score MLM).
3. **Calcul de similarité sémantique :** Pour chaque mot prédit par le MLM, son embedding est généré à l'aide du modèle Sentence Transformer ‘all-MiniLM-L6-v2‘. Simultanément, un embedding du mot cible original est également calculé. La similarité cosinus est ensuite utilisée pour quantifier la proximité sémantique entre le mot original et chacun des mots prédits.
4. **Classement des résultats :** Les mots prédits sont enfin classés par ordre décroissant de leur score de similarité sémantique avec le mot cible original. Cela permet d'identifier les termes les plus proches sémantiquement, indépendamment de leur probabilité de prédiction directe par le MLM.

Exemple de déroulement :

Prenons la phrase "Today, I get the result of my blood tests that revealed the presence of a specific **chemical** biomarker." avec "chemical" comme mot cible.

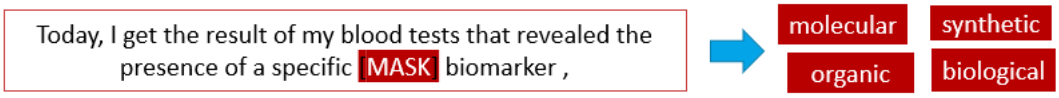


Figure 3 – Exemple de prédiction de model bert

— La phrase est d'abord masquée : "Today, I get the result of my blood tests that revealed the presence of a specific `[MASK]` biomarker."
— Le modèle BERT prédit les mots les plus probables pour remplir le masque, tels que "biological", "molecular", "synthetic", "organic", etc., chacun avec son score MLM.
— Ensuite, pour chacun de ces mots prédits, leur similarité sémantique avec le mot "chemical" est calculée. Par exemple, "molecular" pourrait avoir une similarité cosinus élevée avec "chemical", même si son score MLM n'est pas le plus élevé.
— Les résultats finaux sont présentés, classés par similarité sémantique, permettant de visualiser les synonymes les plus pertinents détectés par le modèle.

Cette approche nous a permis d'obtenir une liste de synonymes potentiels pour chaque type d'entité, enrichissant notre compréhension des termes contextuellement liés dans le domaine biomédical, comme présenté dans la tableau 6.1.5. Les résultats de cette étape servent dans les évaluations de GLiNER .

6.1.2 Pipeline de transformation

Les fichiers texte et leurs annotations associées sont d'abord fusionnés, puis convertis en quatre formats JSON distincts :

Fusion texte–annotation Chaque résumé (`.txt`) possède un identifiant unique, `text_id`, qui figure aussi dans son fichier d'annotation `.ann`. La fusion s'effectue donc en deux temps :

1. **Appariement par `text_id`** Pour chaque `text_id` rencontré :
   — le contenu du fichier `<text_id>.txt` est chargé ;
   — son homologue `<text_id>.ann` est ouvert et parsé.
2. **Construction du couple** On crée une entrée structurée où le texte brut est immédiatement suivi de la liste de ses entités :

$$\text{couple} = \{\ \texttt{text\_id},\ \texttt{text},\ \texttt{entities}\ \}$$

   Cette structure garantit qu'un même identifiant référence toujours le bon duo texte–annotation . Ensuite , on construit quatre fichier (.json)

1. deux fichier pour l'évaluation, conservant le texte brut et ses annotations(une version en code d'entitée et l'autre en defenition d'entitée) .
   — *eval_code.json* : texte brut + code d'entités (`DISO`, `CHEM`, etc.) ;
   — *eval_def.json* : même texte, mais chaque code d'entitées est remplacé par sa *définition* .
2. deux fichier pour le fine-tuning, où le texte est tokenisé et les codes d'entités ne sont plus pris en compte mais en prend l'entité lui même détectée dans la document . Dans l'autre fichier est de remplacés le code d'entitée par leur définition .
   — *train_code.json* : text Tokenisé + les entités détecte dans le text +spans ;
   — *train_def.json* : text Tokenisé + les définitions des codes d'entitée +spans

6.1.3 Format du fichier d'évaluation

Chaque document est structuré de la manière suivante : le "text-id" utilisé comme indice de référence avec la quelle on peut par la suite compare les prédiction de GLiNER avec la veritée terrain , le text présent le cible de GLiNER , et les "entity" et le "spans"

représentent la veritée terrain .

Listing 1 – extrait de la courpus d'évaluation

```
{
  "text_id": "26355933",
  "text": "Experience with cryopreserved homografts ...",
  "entities": [
    {
      "code_entity": "DEVICE",
      "spans": [31, 41],
      "entity": "homografts"
    }
  ]
}
```

Listing 2 – extrait de la courpus d'évaluation

```
{
  "text_id": "26355933",
  "text": "Experience with cryopreserved homografts ...",
  "entities": [
    {
      "code_entity": "Manufactured object used for medical or
         laboratory purposes.",
      "spans": [31, 41],
      "entity": "homografts"
    }
  ]
}
```

6.1.4 Format du fichier de fine-tuning

Pour la fine tuning j'ai conçus deux variantes :

— Première variante avec les codes des entités (`DISO`, `CHEM`, etc.) ;

Listing 3 – Fine-tuning avec code d'entité

```
{
  "tokenized_text": ["AIM", "to", "compare", "tear", ","],
  "ner": [
    ["DISO", 6, 26]
  ]
}
```

— deuxième variante : En remplaçant les codes des entités par ses définitions complètes .

Listing 4 – Fine-tuning avec définition complète

```
{
  "tokenized_text": ["AIM", "to", "compare", "tear", ","],
  "ner": [
    ["Any deviation from the normal state of an organism:
       diseases, symptoms",[ 6, 26]
  ]
}
```

6.1.5 Répartition des données

La table suivante présente les différents codes d'entités annotés dans notre corpus, avec pour chacun : le nombre d'occurrences, sa proportion relative (en pourcentage), ainsi que les synonymes associés qui etait déterminée par la méthodologie dans la partie **??** .

Code d'entité	Occurrences	Distribution (%)	Synonymes associés
DISO	2245	30.5 %	*disease*, *disorder*, *syndrome*, *pathology*
ANATOMY	1824	24.9 %	*organ*, *body part*
CHEM	1161	15.9 %	*chemical*, *compound*, *substance*, *medicati*
FINDING	810	11.1 %	*finding*, *observation*, *result*
PHYS	784	10.7 %	*physiology*, *biological process*, *bodily func*
LABPROC	355	4.7 %	*procedure*, *test*, *examination*
INJURY_POISONING	111	1.5 %	*injury*, *poisoning*, *tension of ligaments*
DEVICE	48	0.7 %	*device*, *apparatus*, *equipment*

Les définitions associées aux entités sont extrait de la base de donnée NEREL-BIO [5]

Code	Définition des entitées
DISO	Any deviation from the normal state of an organism : diseases, symptoms, dysfunctions, organ abnormalities (excluding injuries or poisoning).
CHEM	Chemical substances, including legal/illegal drugs and biomolecules.
DEVICE	Manufactured object used for medical or laboratory purposes.
LABPROC	Testing of body substances and other diagnostic procedures such as ultrasonography.
PHYS	Biological function or process in an organism, including organism attributes (e.g. tempera excluding mental processes.
ANATOMY	Organs, body parts, cells, cellular components and body substances.
FINDING	Statement conveying the results of a scientific observation or experiment.

Table 3 – Correspondance codes → définitions

Enfin , pour la fine tuning j'ai fait un repartition de 90 % pour l'entraînement et 10% pour le test

[h]

	Entraînement (90 %)	**Évaluation (10 %)**
Documents	93	11
Tokens	27 049	3 676
Entités annotées	6 467	827

6.2 Procédure d'évaluation

6.2.1 Introduction

L'objectif de cette section est de mesurer les performances du modèle **GLiNER** dans la reconnaissance d'entités biomédicales au sein des documents scientifiques. L'évaluation repose sur un jeu de données annoté manuellement, issu du corpus *BioASQ*, utilisé comme référence (vérité terrain).

Je vais introduire deux voie d'évaluation du modele GLiNER , la 1 ére évaluation sera fais sur la choix des synonyms pour etudie le taux de recouvrement entre les synonyms de même code d'entité entre elle pour qu'on savoir les quelles sont le plus contributifs à la reconnaissance d'entités et apport à GLiNER plus d'information , l'évaluation seras suit le schéma suivant :

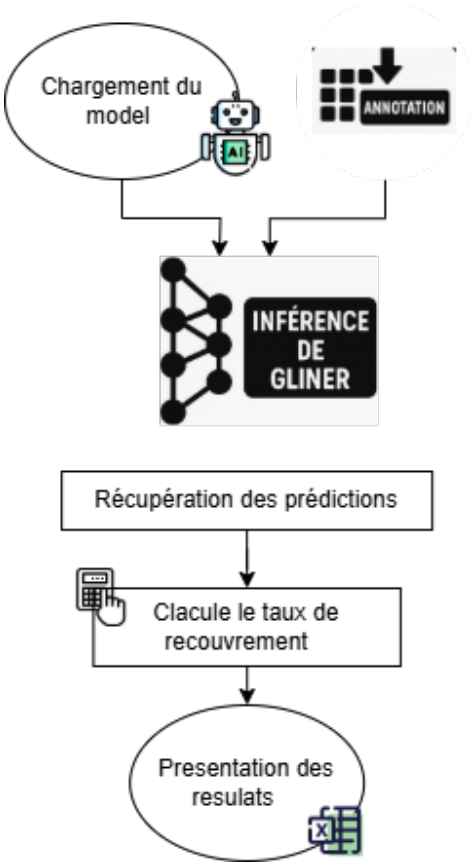


Figure 4 – Architecture d'évaluation des taux de recouvrement entre les synonyms .

la 2 éme évaluation sera à évalue la performance de GLiNER en comparant ses prédiction à les annotation de veritée terrain viens des donnée sources annotée pour justifier l'efficacité de GLiNER par des différent métrique classique et de matching ,

comme explique dans le schéma suivantes :

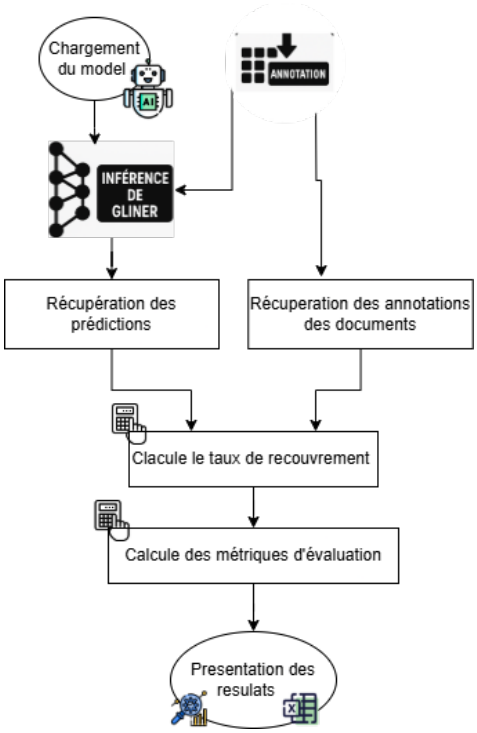


Figure 5 – Architecture d'évaluation de GLiNER via différent métrique .

Par la suite je vais detaille la protocole d'évaluation , comme vous pouvez remarque dans les sachems ci-dessus le deux première étapes sont commun donc on le conserver pour le deux pipeline la même explication ensuite en entre dans le détaille de chaque approche .En commençant par la 1 ére étapes

6.2.2 Étape 1 : Chargement du modèle et préparation des données

La première étape du pipeline d'évaluation consiste à charger le modèle **GLiNER** en mémoire et à activer, si disponible, l'accélération GPU pour optimiser les temps d'inférence.

Les entités annotées dans le corpus sont représentées par des codes (`DISO`, `CHEM`, etc.). Afin d'enrichir la capacité du modèle à détecter des expressions variées pour une même entité, nous associons à chaque code un ensemble de *synonymes* déterminée comme décrit dans la section 5.1.1**??** . Ces synonymes sont fournis au modèle comme étiquettes d'entrée (input labels) pour améliorer sa couverture sémantique.

Simultanément, les documents sont chargés depuis un fichier JSON structuré que j'ai decire en section 6 .

Le texte est utilisé comme entrée pour le modèle, tandis que les annotations servent de vérité de terrain (ground truth) pour le calcul des métriques d'évaluation (précision, rappel, F1-score, etc.).

À l'issue de cette étape, le modèle est prêt à effectuer l'inférence sur chaque document, et les informations nécessaires à l'évaluation (texte + annotations) sont disponibles .

6.2.3 Étape 2 : Inférence GLiNER

Cette étape consiste à interroger le modèle GLiNER sur un texte donné pour extraire les entités selon des labels définis. Nous détaillons ci-dessous chaque phase de l'inférence, puis présentons un algorithme pseudocode montrant comment l'intégrer dans un pipeline d'évaluation.

2.1 – Définition des labels On commence par spécifier la liste des *libellés* (labels) correspondant aux types d'entités que l'on souhaite détecter :

$$\texttt{labels} = [\text{“disease”},\ \text{“syndrome”},\ \ldots]$$

Chaque label sert de consigne textuelle pour guider l'extraction.

2.2 – Construction d'Input Le modèle attend un format unifié où tous les labels sont concaténés en amont du texte à analyser, séparés par un token spécial `[ENT]`, suivi d'un délimiteur `[SEP]`. Par exemple :

$$\underbrace{\texttt{[ENT] disease [ENT] chemical}}_{\text{labels}}\ \texttt{[SEP]}\ \underbrace{\langle\text{texte}\rangle}_{\text{document}}$$

Cette formulation permet au GLiNER d'encoder conjointement la description des entités et le contexte.

2.3 – Appel à la fonction d'inférence Une fois le prompt constitué, on invoque la méthode `predict_entities` du modèle :

```
entities = model.predict_entities(
    text,               # Cible de GLiNER
    labels,             # liste de labels comme ci-dessus
    threshold=0.5  # seuil sur la probabilité sigmoid pour filtrer
)
```

— `threshold` fixe la valeur minimale de $\varphi(i, j, t)$ (score sigmoïde) pour qu'un candidat soit retenu comme entité.

2.4 – Format de la sortie La méthode renvoie une liste de dictionnaires, chacun décrivant une entité extraite :

```
[
  {'text':'Alzheimer disease','label':'disease','start':21,'end':39},
  {'text':'Ibuprofen','label':'chemical','start':81,'end':90}
]
```

où `start` et `end` sont des indices character-level dans le texte original.

Algorithme 1 : Inférence avec GLiNER

Algorithm 1 Inférence d'entités avec GLiNER

Input: texte : `str`, labels : $[\ell_1, \ldots, \ell_M]$, modèle $\mathcal{M}$

Output: Liste d'entités $\mathcal{E}$

`// 1. Préparer le prompt`

`prompt` $\leftarrow$) *concatenate*$\Big([\texttt{[ENT]}, \ell_1, \texttt{[ENT]}, \ell_2, \ldots, \ell_M, \texttt{[SEP]}, \texttt{texte}]\Big)$

`// // 2. Appel de l'inférence`

$\mathcal{R} \leftarrow \mathcal{M}$.`predict_entities`(`texte`, $[\ell_1, \ldots, \ell_M]$, `threshold`)

`// // 3. Filtrer et formater les résultats`

$\mathcal{E} \leftarrow []$

foreach $r \in \mathcal{R}$ **do**

  if r[`probability`] $\geq$ *threshold* **then**

    Ajouter (r[`text`], r[`label`], r[`start`], r[`end`]) à $\mathcal{E}$

return $\mathcal{E}$

Ainsi, cette étape permet de transformer un texte brut et une liste de labels en un ensemble des prédictions de spans labellisés, prêtes pour l'étape suivante d'alignement et d'évaluation.

6.2.4 Étape 3 : la premier pipeline d'évaluation : calcule d'indice de Jaccard :

On récupérer les ensembles prédictions de GLiNER pour chaque synonyms d'entitée et on calcule l'indice de Jaccard sur ces ensemble deux par deux :

$$\text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|},$$

où A et B sont les ensembles de positions caractères (spans) des prédiction de deux synonymes de même code d'entitée.Le but ici est de mesure le taux de chevauchement entre les différent synonymes s'il y a des synonymes qui sont déjà couvertes par des autres synonymes en terme de champs de prédiction par GLiNER .

6.2.5 Étape 3 : La deuxième pipeline d'évaluation :

Cet étape a pour objectif de faire correspondre les entités prédit par GLiNER et les entités annotées dans le corpus (la «vérité terrain»), afin de calculer les quatre

compteurs fondamentaux :

$$\{TP,\ FP,\ FN,\ TN\}$$

Ces compteurs serviront de base au calcul de toutes les métriques d'évaluation (précision, rappel, F1, etc...).

En première lieux en va procède en filtrage par label,pour chaque entité prédite p de code t_p, on ne la compare qu'avec les annotations de vérité terrain ayant le même code $t_g = t_p$.

Ainsi ,ce filtrage assure que seules les entités du même type sont mises en correspondance, évitant les confusions inter-catégories.

Il exist deux cas de matching ,pour chaque prédiction s_p et chaque annotation s_g :

— *Exact match* : on considère la prédiction comme correcte si

$$s_p = s_g \quad \text{et} \quad t_p = t_g.$$

— *Partial match / Indice de Jaccard* : on calcule

$$\text{Jaccard}(s_p, s_g) = \frac{|s_p \cap s_g|}{|s_p \cup s_g|},$$

où s_p et s_g sont les ensembles des positions caractères respectivement de l'ensemble de prédiction et de la veritée terrain . Donc Si

$$\text{Jaccard}(s_p, s_g) \geq 0{,}5 \quad \text{et} \quad t_p = t_g,$$

on considère la correspondance partielle est validée par la suite cette prédiction est compte aussi comme TP .

Après qu'on calcule l'exact match et l'indice de Jaccard pour tous les prédictions de GLiNER on passe à l'**attribution des statuts** pour chaque prédiction de GLiNER

— On parcourt d'abord toutes les prédictions fais sur notre ensemble des labels . On appliquant les critères ci-dessus pour chaque spans s_p et s_g , et on marque p comme **TP** si l'un de critère est validée et on consomme g pour qu'elle ne soit plus disponible pour une autre prédiction.
— Les prédictions restantes non appariées sont classées **FP**.
— Enfin, toutes les annotations g de vérité terrain qui n'ont pas été appariées deviennent **FN**.
— On peut déduire les **TN** comme les intervalles non couverts ni par une prédiction ni par une annotation.

Implémentation Algorithmique

Algorithm 2 Évaluation des entités prédictes par GLiNER

Input:

P : Liste des entités prédites $p_i = (s_p, t_p)$

G : Liste des entités annotées (vérité terrain) $g_j = (s_g, t_g)$

Output:

Attribution des statuts : TP, FP, FN

Comptage des *exact match* et *partial match*

Initialisation :

Marquer toutes les annotations $g_j \in G$ comme non appariées.

Définir des compteurs : TP, FP, $FN \leftarrow 0$

foreach *entité prédite* $p = (s_p, t_p) \in P$ **do**

  $matché \leftarrow$ False

  foreach *annotation* $g = (s_g, t_g) \in G$ *non encore appariée* **do**

    if $t_p = t_g$ **then**

      if $s_p = s_g$ **then**

        (Exact Match)

        Marquer p comme **TP** et g comme appariée

        Incrémenter TP, enregistrer comme exact match

        $matché \leftarrow$ True ; **break**

      else

        Calculer $J = \text{Jaccard}(s_p, s_g)$

        if $J \geq 0.5$ **then**

          (Partial Match)

          Marquer p comme **TP** et g comme appariée

          Incrémenter TP, enregistrer comme partial match

          $matché \leftarrow$ True ; **break**

  if $matché == False$ **then**

    Marquer p comme **FP**

    Incrémenter FP

foreach *annotation* $g \in G$ *non appariée* **do**

  Marquer g comme **FN**

  Incrémenter FN

à la fin de cette étapes on aura un file json contiens les entities prédit par GLiNER avec ses statues :

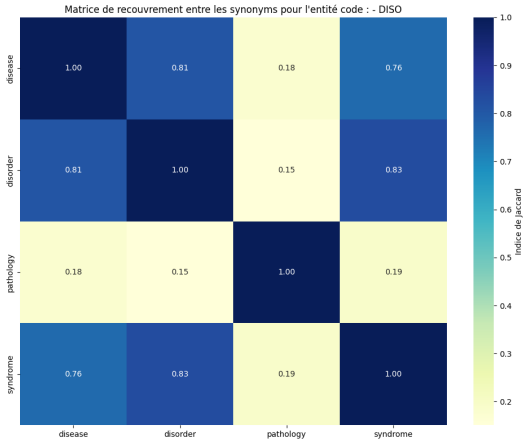
Listing 5 – extrait de la courpus d'évaluation


```
{
  "text_id": "25909557",
  "status": "FP",
  "span": [
    134,
    157
  ],
  "entity_text": "liver metastatic lesion",
  "predicted_entity": "liver metastatic lesion",
  "entity_code": "INJURY_POISONING",
}
```

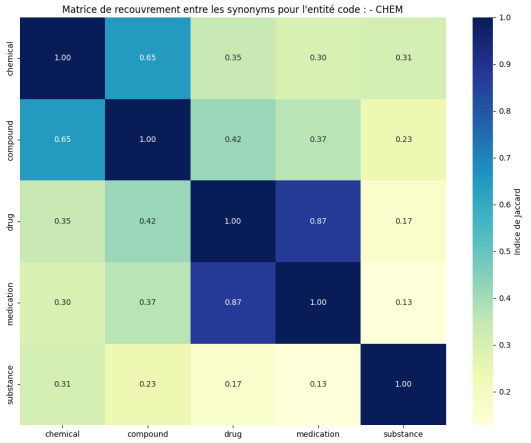

7 Résultats et interprétation

7.1 Analyse des résultats de Première évaluation

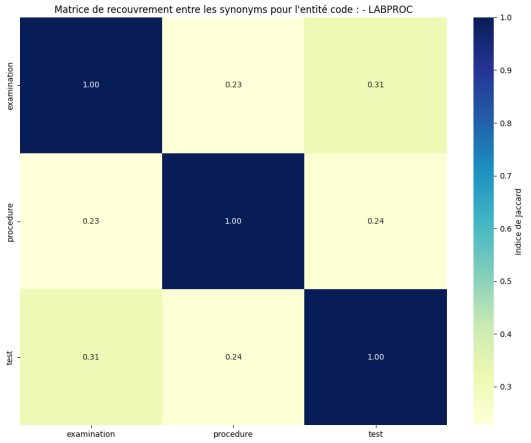
Dans le cadre de l'évaluation du modèle GLiNER,j'ai analysé le chevauchement des ensembles prédite selon les différents synonymes utilisés pour chaque type d'entité. Pour ce faire, nous avons réalisée les heatmap des indices de Jaccard entre les ensembles prédit associées à chaque paire de synonymes d'une même entité, comme décrit dans la partie 7.4 .



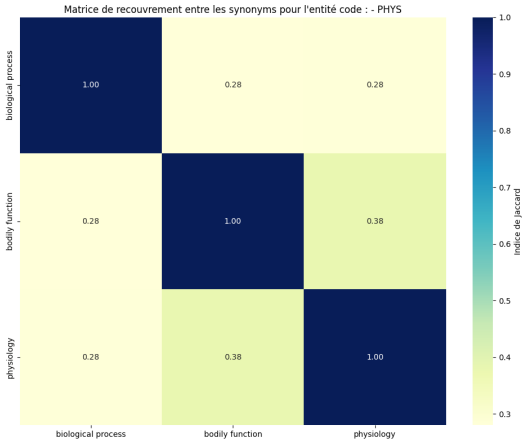
(a) DISO



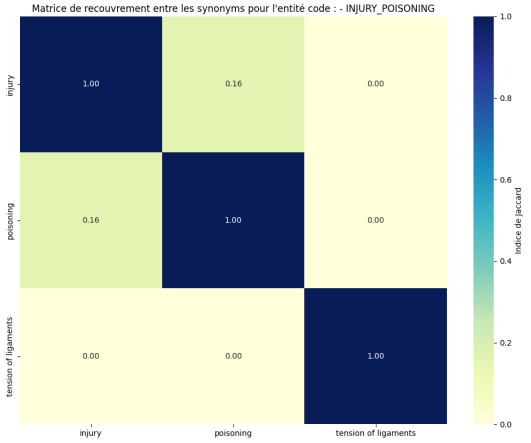
(b) CHEM



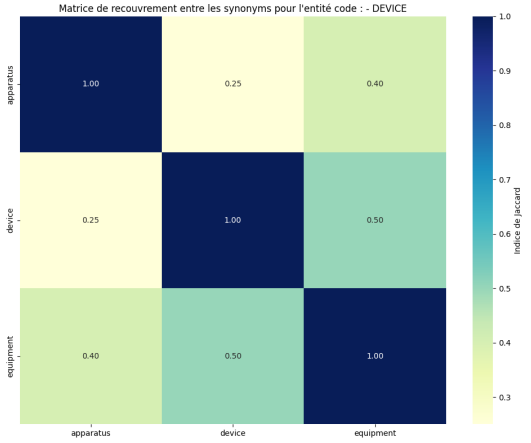
(c) LABPROC



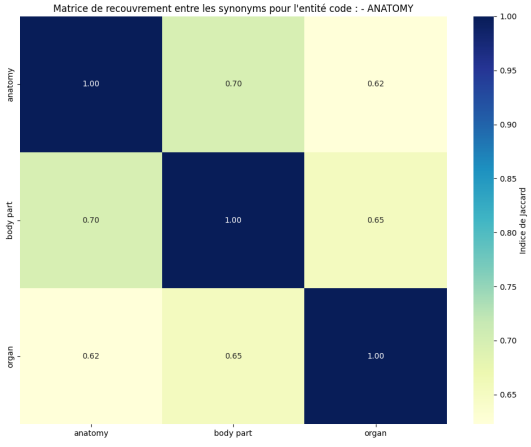
(d) PHYS



(e) INJURY_POISONING



(f) DEVICE



(g) ANATOMY

Interprétation : Le but de cette analyse est de mesurer le taux de chevauchement entre les prédictions des différents synonymes. Un Jaccard élevé indique que deux synonymes conduisent à des prédictions très similaires, suggérant une redondance .Tandis qu'un Jaccard faible, au contraire, indique que les synonymes capturent des entités différentes, suggérant une complémentarité entre eux .
On remarque un fort Chevauchement entre drug et medication par valeur de 0.87.Cela indique que les prédictions faites par GLiNER lorsqu'il recherche 'drug' sont très similaires à celles faites lorsqu'il recherche 'medication'. Ces deux termes capturent quasiment les mêmes entités "CHEMICAL".

En outre on trouve un Chevauchement Modéré entre chemical et compound de de 0.65 qui représente un chevauchement substantiel. Bien qu'ils ne soient pas aussi interchangeables que 'drug' et 'medication', une part importante de leurs prédictions se recoupe.

En fin les plus robuste synonymes sont les uns qui présent des faibles Chevauchements avec les autres car on ne chercher pas à mettre les synonymes en compétition mais plus tôt à couvrir le plus vaste possible champs textuelle comme substance qui à des valeur relativement faible :

- `substance` vs `medication` : 0.13
- `substance` vs `drug` : 0.17
- `substance` vs `compound` : 0.23
- `substance` vs `chemical` : 0.31

Cela signifie que `substance` identifie un ensemble d'entités "CHEMICAL" largement différent de celui détecté par les autres synonymes.

Cette analyse suggère que le choix des synonymes dans les entrées du modèle peut significativement affecter les prédictions. Une choix diversifie dans les synonymes (label) pourrait consister à améloire les résultats issus de plusieurs synonymes,par la suite la performance du modéle GLiNER sur la task du NER.

7.2 Analyse Quantitative de la Performance du Modèle GLiNER

Cette section est dédiée à l'évaluation quantitative des performances du modèle GLiNER, un aspect crucial pour mesurer l'efficacité de notre approche face à la diversité des termes et des contextes spécifiques du domaine biomédical. Notre objectif est de déterminer dans quelle mesure GLiNER, en s'appuyant sur les listes de synonymes fournies pour chaque type d'entité, est capable de reconnaître et de catégoriser correctement les informations pertinentes. Pour ce faire, notre évaluation se décompose en trois axes principaux, chacun offrant une perspective complémentaire sur la performance du modèle

7.2.1 Présentation des résultats de deuxième évaluation

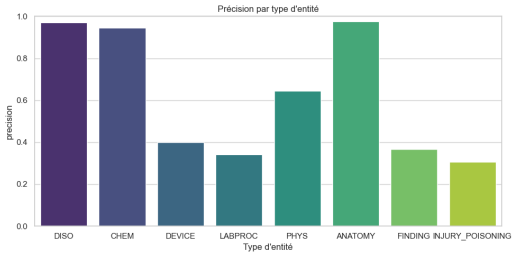
Premièrement, nous procéderons à une évaluation agrégée sur l'ensemble des types d'entités. Cette étape initiale consistera à évaluer globalement la performance de GLiNER pour chacun des huit types d'entités que nous avons définis, en consolidant les prédictions issues de tous leurs synonymes respectifs. Cette vue macroscopique servira de fondation à notre analyse.

Deuxièmement, nous concentrant sur l'évaluation des performances par les synonymes individuels pour le type d'entité 'CHIMIQUE'. Nous examinerons la contribution de chaque synonyme pris isolément à la détection des entités, afin d'identifier les termes les plus pertinents et ceux qui pourraient introduire de l'ambiguïté ou du bruit.

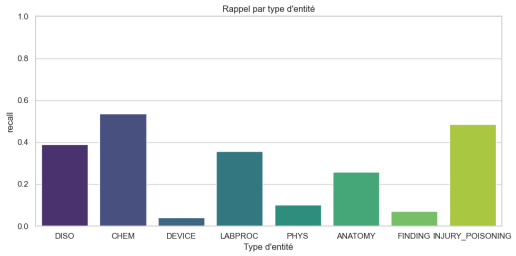
Enfin, nous explorerons l'influence des stratégies de combinaison des prédictions des synonymes. Pour le type d'entité 'CHIMIQUE', particulièrement représentatif, nous présenterons une analyse détaillée des résultats obtenus en considérant à la fois l'intersection et l'union des ensembles d'entités prédites par les différents synonymes.

Alors on commençant par :

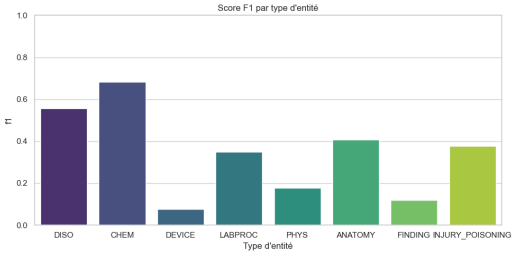
— Évaluation agrégée sur l'ensemble des types d'entités



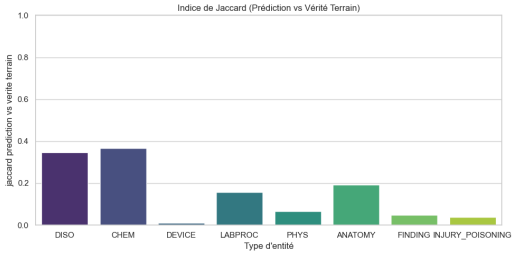
(a) Précision par type d'entité



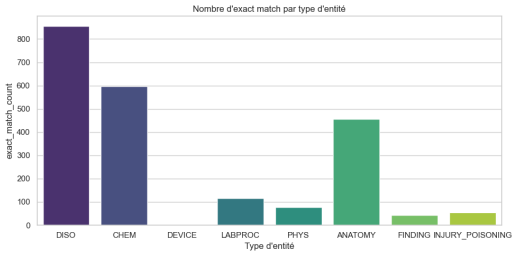
(b) Recall par type d'entité



(c) Score F1 par type d'entité



(d) Indice de Jaccard vs Vérité terrain



(e) Nombre de correspondances exactes

Figure 7 – Évaluation des performances de GLiNER selon différentes métriques par type d'entité

— Évaluation individuelle des synonymes (de CHEMICAL) et des stratégies de combinaison de synonymes .

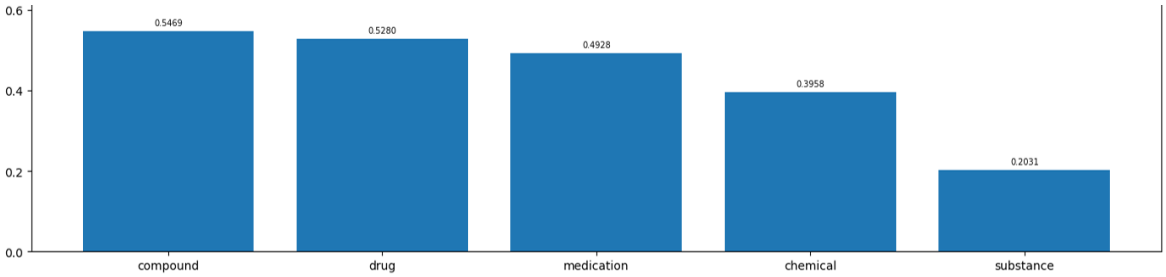
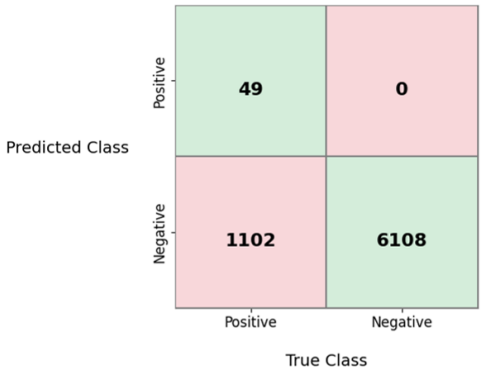
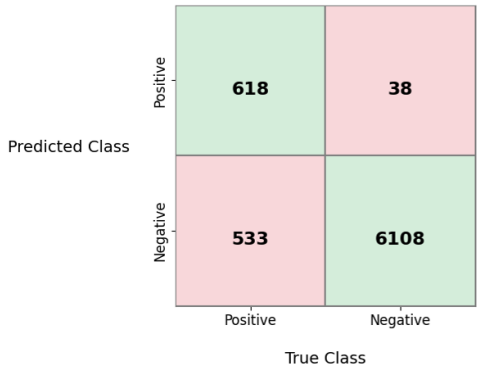


Figure 8 – Performance (F1-score) du GLiNER pour chaque synonyme individuel associé au même type d'entité *CHEMICAL*.



(a) Matrice de confusion basée sur l'intersection des entités extraites.



(b) Matrice de confusion basée sur l'union des entités extraites.

Figure 9 – Matrices de confusion pour la détection des entités *CHEMICAL*, en fonction de la stratégie de combinaison des synonymes.

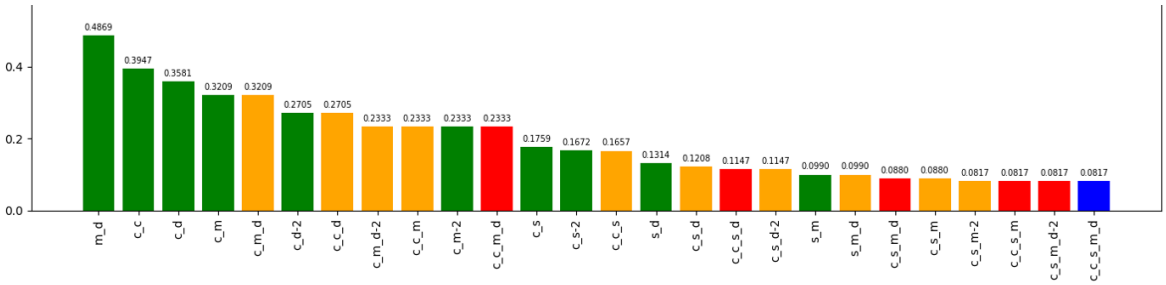


Figure 10 – Performance (F1-score) pour le type d'entités *CHEMICAL*, montrant l'influence des combinaisons de synonymes via l'intersection des ensembles d'entités extraites.

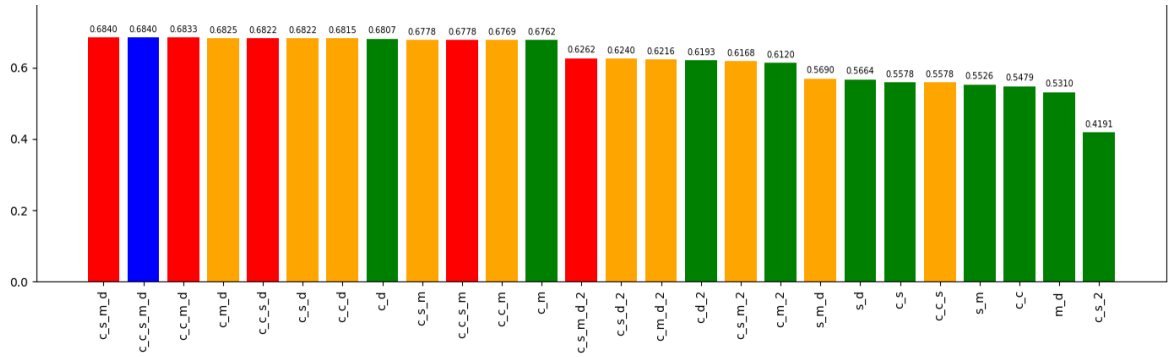


Figure 11 – Performance (F1-score) pour le type d'entités *CHEMICAL*, montrant l'influence des combinaisons de synonymes via l'union des ensembles d'entités extraites.

Légende des synonymes : M :chemical _ C :compound _ S :substance _ M-2 :medication _ D :drug.

7.2.2 Analyse de performance avec les Synonymes Individuels (Figure 8)

La Figure 8 présente le Score F1 pour l'extraction d'entités "CHIMIQUE" en considérant chaque synonyme individuellement. En effet on a basé notre analyse sur le score F1 car il est le plus utilisée en cas des données déséquilibrées (notre cas) ,il offre une vue plus juste des performances du modèle en considérant à la fois la capacité à prédire correctement les classes positives (précision) et à identifier toutes les instances positives (rappel), là où l'exactitude seule pourrait être trompeuse . Un Score F1 plus élevé indique un meilleur équilibre entre précision et rappel pour le synonyme donné.

Table 4 – Performance (Score F1) par Synonyme Individuel

Synonyme	Score F1
compound	0.5469
drug	0.5280
medication	0.4928
chemical	0.3958
substance	0.2031

Observations :

— Le modèle fonctionne le mieux lorsque 'compound' est utilisé comme synonyme, atteignant un Score F1 de 0.5469. Cela suggère que le modèle est efficace pour identifier les entités CHIMIQUE lorsqu'elles sont désignées par 'compound'.

— 'Drug' montre également une forte performance, proche de 'compound', avec un Score F1 de 0.5280.
— 'Medication' suit avec un Score F1 modéré de 0.4928.
— Le terme de base 'chemical' donne un Score F1 inférieur (0.3958) comparé à 'compound', 'drug' et 'medication'. Cela pourrait indiquer que 'chemical' est un terme plus ambigu dans les données d'évaluation ou que son contexte est plus difficile à capturer précisément pour le modèle.
— 'Substance' présente la performance la plus basse avec un Score F1 de 0.2031. Cela suggère que se fier uniquement à 'substance' pour l'extraction d'entités CHIMIQUE n'est pas efficace, probablement en raison de sa nature large et générique, conduisant à de nombreux faux positifs ou faux négatifs.

Implication : Lors de l'inférence pour des synonymes individuels, 'compound' et 'drug' sont les termes les plus efficaces pour identifier les entités "CHIMIQUE" avec ce modèle GLiNER .

7.2.3 Influence des Combinaisons de Synonymes (Figure 10/11)

La Figure 10 offre une vue complète de l'impact des combinaisons de synonymes sur le Score F1, en distinguant entre l'intersection et l'union des ensembles d'entités extraites.

Diagramme à barres :Intersection des ensembles d'entités extraites Ce diagramme montre le Score F1 lorsque seules les entités identifiées par *tous* les synonymes d'une combinaison donnée sont prises en compte. Cette approche met l'accent sur la précision, visant à n'identifier que les extractions très fiables.

Observations :

— Les Scores F1 sont généralement plus bas pour les ensembles d'intersection par rapport aux performances des synonymes individuels. Ceci est attendu, car exiger que plusieurs synonymes s'accordent sur une extraction réduira le nombre d'entités identifiées (augmentant potentiellement la précision mais diminuant souvent le rappel).
— Le Score F1 le plus élevé (environ 0.4669) est atteint pour 'c_s_m_d' (compound, substance, medication, drug), ce qui implique que lorsque ces quatre termes s'accordent collectivement sur une entité, l'extraction est raisonnablement fiable.
— Les combinaisons impliquant 'substance' (par exemple, 's_d', 's_m', 'c_s', 'c_s_d', 'c_s_m') apparaissent souvent dans la partie inférieure des Scores F1, renforçant l'observation de la Figure 4 selon laquelle 'substance' est un indicateur plus faible.
— Les Scores F1 globalement bas dans le graphique d'intersection indiquent que le fait de se fier à l'intersection de plusieurs extractions de synonymes

pourrait conduire à une précision élevée mais à un rappel considérablement réduit, manquant potentiellement de nombreux vrais positifs. Cette stratégie est appropriée lorsque les faux positifs sont extrêmement coûteux.

Diagramme à barres : Union des ensembles d'entités extraites

Ce diagramme montre le Score F1 lorsque toutes les entités identifiées par *n'importe quel* synonyme d'une combinaison donnée sont prises en compte. Cette approche priorise le rappel, visant à capturer autant de vrais positifs que possible.

Observations :

— Les Scores F1 pour les ensembles d'union sont nettement plus élevés que pour les synonymes individuels ou les ensembles d'intersection, indiquant une bien meilleure performance globale. Cela suggère que la combinaison des sorties de différents synonymes en prenant leur union améliore considérablement la capacité du modèle à identifier les entités .

— De nombreuses combinaisons atteignent des Scores F1 supérieurs à 0.6, les plus élevés atteignant environ 0.6840 (pour 'c_s_m_d' et 'c_s_c_m_d').

— Les combinaisons les plus performantes ('c_s_m_d', 'c_s_c_m_d', 'c_m_d', 'c_c_d', 'c_s_d', 'c_c_m') incluent souvent 'compound', 'drug' et 'medication', qui étaient individuellement des termes performants.

— Même les combinaisons incluant 'substance' ('c_s_m_d', 'c_s_d') fonctionnent bien dans le scénario d'union, démontrant que, bien que 'substance' soit faible seul, son inclusion dans une union avec des synonymes plus forts peut contribuer positivement au rappel global sans nuire excessivement à la précision.

Implication : La combinaison des extractions de plusieurs synonymes en prenant leur union est la stratégie la plus efficace pour maximiser le Score F1 pour l'extraction des entités avec le modèle Gliner. Cela suggère que différents synonymes capturent différentes facettes de l'entité , et leur couverture combinée est plus complète.

7.2.4 Matrices de Confusion pour les Combinaisons de Synonymes (Figure 9)

La Figure 9 fournit les nombres bruts de True Positive (TP), False Positive (FP), True Negative (TN) et False Negative (FN) pour le type d'entité "CHIMIQUE" dans deux scénarios : intersection et union des ensembles d'entités extraites.

7.2.5 Matrice de Confusion d'Intersection des ensembles d'entités extraites

Table 5 – Matrice de Confusion - Intersection des Ensembles

	Vrai Positif	**Vrai Négatif**
Classe Prédite : Positive	49 (TP)	0 (FP)
Classe Prédite : Négative	1102 (FN)	6108 (TN)

Calculs des métriques :

— **Total des Instances** : $49 + 0 + 1102 + 6108 = 7259$

— **Précision** : $\frac{49}{49+0} = 1$ (100%)

— **Rappel** : $\frac{49}{49+1102} = \frac{49}{1151} \approx 0.0426$ (4.26%)

— **Score F1** : $2 \times \frac{1 \times 0.0426}{1+0.0426} \approx 0.0818$

— **Exactitude** : $\frac{49+6108}{7259} \approx 0.8481$ (84.81%)

Observations :

— **Précision Extrêmement Élevée (100%)** : L'observation la plus frappante est la précision parfaite. Cela signifie que chaque fois que le modèle prédit une entité "CHIMIQUE" dans le scénario d'intersection, il est toujours correct (0 Faux Positif). Cela est souhaitable lorsque les faux positifs sont très indésirables.

— **Très Faible Rappel (4.26%)** : Cependant, cela se fait au prix d'un rappel significativement faible. Le modèle n'identifie que 49 entités "CHIMIQUE" réelles sur 1151. Cela signifie qu'il manque une grande majorité (1102) des vraies entités "CHIMIQUE".

— **Nombre Élevé de Vrais Négatifs** : Le nombre élevé de Vrais Négatifs (6108) indique que le modèle est très bon pour identifier correctement les instances qui ne sont *pas* des entités "CHIMIQUE".

Interprétation : Cette approche d'évaluation par intersection met en lumière le comportement le plus conservateur et restrictif de GLiNER. Le résultat observé est en effet attendu : pour qu'une entité soit détectée et validée dans l'ensemble d'intersection, elle doit impérativement être reconnue et confirmée par l'ensemble de tous les synonymes fournis pour cette entité. Ce processus de validation croisée multi-synonymes rend les prédictions extrêmement précises (augmentant la précision), mais limite drastiquement le champ des détections possibles par le modèle, conduisant inévitablement à un rappel très faible."

En pratique, un tel modèle ne serait approprié que dans des scénarios très spécialisés où même un seul faux positif est inacceptable et où manquer un grand nombre de vrais positifs est une conséquence tolérable ou même souhaitée.

Pour la plupart des tâches de NER , où un équilibre entre précision et rappel est recherché, cette approche par intersection pure n'est pas optimale, car elle sacrifie trop de couverture pour une validation excessive.

7.2.6 Matrice de Confusion d'Union des ensembles d'entités extraites

Table 6 – Matrice de Confusion - Union des Ensembles

	Vrai Positif	**Vrai Négatif**
Classe Prédite : Positive	618 (VP)	38 (FP)
Classe Prédite : Négative	533 (FN)	6108 (VN)

Calculs des métriques :

— **Total des Instances** : $618 + 38 + 533 + 6108 = 7297$

— **Précision** : $\frac{618}{618+38} = \frac{618}{656} \approx 0.9421$ (94.21%)

— **Rappel** : $\frac{618}{618+533} = \frac{618}{1151} \approx 0.5370$ (53.70%)

— **Score F1** : $2 \times \frac{0.9421 \times 0.5370}{0.9421+0.5370} \approx 0.6865$

— **Exactitude** : $\frac{618+6108}{7297} \approx 0.9217$ (92.17%)

Observations :

— **Performance Équilibrée** : Comparée à l'intersection, l'approche par union offre une performance beaucoup plus équilibrée.

— **Précision Élevée (94.21%)** : La précision reste très élevée, ce qui signifie que lorsque le modèle prédit une entité "CHIMIQUE" en utilisant l'approche par union, il est correct plus de 94% du temps. Seuls 38 Faux Positifs sont observés.

— **Rappel Significativement Amélioré (53.70%)** : Le rappel augmente considérablement, passant de 4.26% à 53.70%. Cela indique que le modèle identifie maintenant plus de la moitié des entités "CHIMIQUE" réelles. Bien qu'il manque encore un nombre substantiel (533 Faux Négatifs), c'est une nette amélioration par rapport à l'intersection.

— **Exactitude Globale** : L'exactitude est également très élevée (92.17%), indiquant une forte performance globale dans la classification correcte des instances positives et négatives.

Interprétation : L'approche par union des ensembles de prédictions de GLiNER, où une entité est validée si elle est reconnue par au moins un synonyme, se révèle être la stratégie la plus pertinente pour évaluer la performance pratique du modèle. Elle permet à GLiNER de tirer parti de la diversité des synonymes fournis pour maximiser la couverture des entités réelles sans compromettre de manière excessive la qualité des prédictions.

Ces résultats sont particulièrement encourageants. Ils suggèrent que GLiNER, même sans entraînement spécifique sur le corpus cible, est capable d'atteindre une performance compétitive pour la reconnaissance d'entités CHIMIQUE. La capacité à maintenir une précision élevée tout en améliorant considérablement le rappel indique que le modèle capte efficacement les relations sémantiques entre les termes et leurs étiquettes. Cela positionne GLiNER comme un outil prometteur pour l'extraction d'informations dans des domaines où l'annotation manuelle est coûteuse et où une stratégie de détection flexible est requise. Le F1-score de 0.6865 représente une base solide, et de futures améliorations pourraient se concentrer sur la réduction des Faux Négatifs restants (533 FN), potentiellement par l'enrichissement des synonymes pour les termes moins bien couverts ou par des techniques de post-traitement des prédictions.

Conclusion et Recommandations

L'analyse quantitative approfondie de la performance du modèle GLiNER a mis en lumière des aspects cruciaux de son comportement et de ses capacités. Les résultats obtenus, en particulier l'influence des synonymes fournis et des méthodes d'agrégation des prédictions (par intersection et par union), sont des indicateurs clés pour comprendre les forces et les lacunes de ce modèle en contexte d'apprentissage zéro-shot.

Les forces majeures de GLiNER résident dans sa robustesse en matière de précision et son efficacité remarquable en apprentissage zéro-shot. Le modèle démontre une capacité impressionnante à maintenir une précision élevée, atteignant 94.21% lors de l'agrégation des prédictions par union. Cette fiabilité des prédictions est un atout considérable, notamment dans des applications où la minimisation des faux positifs est primordiale. L'analyse par intersection a d'ailleurs révélé que GLiNER est capable de générer des prédictions d'une précision parfaite (100%) lorsque les critères de consensus entre les synonymes sont les plus stricts, validant ainsi la qualité intrinsèque des signaux sémantiques qu'il peut extraire. Plus globalement, les scores F1 obtenus, avec un 0.6865 prometteur pour l'union des synonymes, sont particulièrement encourageants. Ils témoignent de la capacité de GLiNER à transférer efficacement les vastes connaissances sémantiques acquises lors de son pré-entraînement généraliste vers une tâche spécifique de reconnaissance d'entités dans un nouveau domaine, et ce, avec un minimum d'effort d'adaptation. Un autre point fort essentiel réside dans le bénéfice clair apporté par la combinaison des synonymes via l'opération d'union. Cette approche a considérablement amélioré la performance globale (F1-score) par rapport à l'utilisation de synonymes individuels ou à l'intersection pure.

Cela confirme que différents synonymes activent des facettes complémentaires de l'entité "CHIMIQUE", et que leur combinaison logique maximise la couverture des entités réelles sans dégrader excessivement la précision. Enfin, l'évaluation individuelle des synonymes a permis d'identifier des termes optimaux comme 'compound' et 'drug', qui se sont avérés les plus efficaces pour déclencher la détection des entités "CHIMIQUE", fournissant ainsi des insights précieux pour l'optimisation future des listes des synonyms .

Malgré ces points forts indéniables, certaines lacunes et points faibles ont été identifiés, principalement liés au rappel et à l'adaptation fine au domaine. La faiblesse la plus notable se manifeste par le rappel extrêmement bas (4.26%) lorsque l'approche d'intersection est appliquée. Bien que ce résultat soit une conséquence logique de la nature hyper-conservatrice de cette méthode, il met en évidence que GLiNER ne parvient à obtenir un consensus de détection unanime que pour une fraction minime des entités réelles. Pour les applications nécessitant une extraction exhaustive, cette configuration n'est clairement pas viable. De plus, même avec l'approche par union, le rappel, bien qu'amélioré (53.70%), reste perfectible avec un nombre substantiel de 533 Faux Négatifs.

Enfin, le modèle présente parfois un décalage taxonomique ou des problèmes de granularité, où ses prédictions (des Faux Positifs) ne correspondent pas précisément aux définitions ou à la portée attendue par notre schéma d'annotation, suggérant une divergence entre les connaissances générales de GLiNER et les spécificités de notre domaine.

Pour maximiser le potentiel de GLiNER dans l'extraction d'informations biomédicales, plusieurs recommandations stratégiques peuvent être formulées. Premièrement, une optimisation et un enrichissement rigoureux des listes de synonymes sont impératifs. Cela implique non seulement l'ajout de termes plus exhaustifs et pertinents pour les entités qui génèrent de nombreux Faux Négatifs, mais aussi l'évaluation critique de l'inclusion ou de la pondération de synonymes trop génériques qui peuvent introduire du bruit ou affaiblir la détection. Deuxièmement, il est crucial d'affiner les définitions des labels utilisées pour GLiNER, en s'assurant qu'elles reflètent précisément les nuances et la granularité de notre schéma d'annotation. Pour les cas de décalage taxonomique (comme 'bleeding' classé différemment de nos attentes), une réévaluation de la cartographie des concepts est nécessaire, potentiellement en utilisant des exemples d'inclusion et d'exclusion pour guider le modèle. Troisièmement, l'implémentation de stratégies de post-traitement peut significativement améliorer la qualité des sorties. Cela pourrait se traduire par la mise en place de filtres pour supprimer des faux positifs connus, l'application de règles de fusion pour reconstituer des entités imbriquée (sous-segmentées multi-tokens), ou des corrections automatiques

de typologie basées sur des dictionnaires ou ontologies spécifiques au domaine. Enfin, d'après mes recherches dans la composant responsable d'attribution des prédictions une meilleur solution sera de recherche un encodeur qui sera plus adapter pour accepter des définitions des code d'entités dans le labels à la place des simples mots et qui sera capable comme j'ai explique dans la partie des méthodologie proposée pour enrichir GLiNER [5] afin d'améliorée ainsi la performance et la fiabilité du modèle dans des contextes cliniques compliquée.

7.3 Analyse Qualitative de la Performance du Modèle GLiNER

Les métriques quantitatives sont essentielles pour évaluer la performance globale d'un modèle ,mais elles ne permettent pas à elles seules de comprendre les raisons sous-jacentes des succès et des échecs. Une analyse qualitative approfondie des erreurs est indispensable pour déceler les schémas comportementaux du modèle et identifier des pistes d'amélioration. Cette section présente une analyse détaillée des Faux Positifs (FP) et Faux Négatifs (FN) predit par GLiNER sur notre corpus, afin de caractériser les types d'erreurs dominantes et d'interpréter le comportement du modèle face à des défis linguistiques et contextuels spécifiques .L'objectif est d'identifier des schémas d'erreurs récurrents, de formuler des hypothèses sur les raisons de ces défaillances et d'éclairer les défis posés par le domaine biomédical

Nous allons examiner les erreurs de GLiNER pour quelques textes en nous basant sur les statuts (FN/FP) et les contexte du text e proposant queleques hypothése à discuter .

Hypothèse 1 : Sur-généralisation du label INJURY POISONING Dans le corpus épileptologique, toutes les occurrences de « seizure aggravation » et assimilés sont classées INJURY POISONING, alors que l'oracle les étiquette DISO. Le même glissement s'observe, dans l'étude chirurgicale, pour « postoperative ventral hernias » et « abdomen ptosis ». La dérive provient d'une heuristique interne : chaque fois qu'un syntagme nominal décrit une altération anatomico-fonctionnelle, GLiNER invoque par défaut la catégorie « lésion », d'où une précision très basse (0,20) pour INJURY POISONING et l'accumulation de FP (206 cas).

Hypothèse 3 : Recouvrement d'entités imbriquées et perte des sous-segments Lorsque GLiNER valide l'entité longue « juvenile myoclonic epilepsy », l'algorithme d'appariement consomme la vérité terrain correspondante puis néglige

les sous-entités « absences », « myoclonic jerks » ou « myoclonic epilepsy ». Ce mécanisme produit simultanément un vrai positif et plusieurs FN, expliquant qu'en dépit d'une haute précision (0,95) le rappel pour DISO s'effondre à 0,42.

Hypothèse 3 : Ambivalence fonctionnelle des termes médicaux communs
Les lemmes fréquents tels que « seizure », « epilepsy », « absences » ou « outcome » adoptent tour à tour le rôle de tête d'entité, de modificateur ou d'anaphore. GLiNER n'opère pas la désambiguïsation contextuelle ; il transforme la majuscule incidente de « Seizures » (début de ligne) en marqueur de résultat clinique (FINDING), tout en omettant le même lemme en minuscule à deux phrases de distance (FN).

Hypothèse 4 : Frontière poreuse entre FINDING et DISO
Dans l'article sur les hernies, « increased physical health » ou « number of excellent results increased » devraient appartenir à FINDING , ils passent pourtant sous le radar du modèle. À l'inverse, les troubles épileptiques sont parfois prédis comme constats isolés. Chaque confusion double l'erreur : elle crée un FP du côté du label choisi et un FN du côté du label attendu, d'où la F-mesure particulièrement faible (0,13) observée pour FINDING.

Hypothèse 5 : Signal trompeur des déverbaux et suffixes en -ment / -ation / -ing « Endoprosthetic replacement » est annoté DEVICE mais systématiquement prédit LABPROC.
Le suffixe -ment évoque une action chirurgicale, poussant GLiNER à catégoriser ces syntagmes comme procédures. En conséquence, LABPROC se surcharge en FP (242) tandis que DEVICE se prive de nombreux TP, ramenant son rappel à environ 0,10.

Hypothèse 6 : Déficit de couverture lexicale pour les matériaux polymères
Les mentions isolées de « polypropylene » ne déclenchent aucun label CHEM ; seule l'expression composée « polypropylene prosthesis » (annotée DEVICE) est reconnue partiellement. La moitié des entités chimiques reste donc manquée (533 FN pour 618 TP), suggérant un manque d'exemples de matériaux simples dans les données d'entraînement.

Hypothèse 7 : Incapacité à gérer les composés techniques de grande longueur
Les chaînes comme « super lightweight polypropylenepolyvinylidenefluoride prosthesis » excèdent la fenêtre de sous-mots du modèle. La segmentation approximative provoque un échec total : ni le matériau (CHEM), ni l'implant

(DEVICE) ne sont capturés. Cette limite structurelle amplifie la perte déjà décrite par l'hypothèse 2.

Hypothèse 8 : Lexique post-opératoire subjectif insuffisamment représenté Les indicateurs de qualité de vie (« physical health », « psychic », « excellent results », « satisfactory outcomes ») constituent un champ lexical sous-échantillonné ; presque toutes ces propositions restent FN. L'écart confirme que GLiNER est surtout sensibilisé à des signes objectivables (biomarqueurs, symptômes physiques) plutôt qu'à des appréciations globales.

Hypothèse 9 : Sensibilité disproportionnée à la casse et aux marqueurs de section La majuscule initiale de « Results », simple titre, déclenche à tort un label FINDING alors qu'aucune entité n'est attendue. Inversement, la minuscule ordinaire en pleine phrase est souvent ignorée. Cette dépendance au style typographique réapparaît pour « Seizures » vs « seizure », consolidant l'idée d'un biais distributionnel.

8 Conclusion

GLiNER démontre une précision remarquable et une capacité de transfert prometteuse en apprentissage zéro-shot, particulièrement lorsqu'une stratégie d'union des synonymes est appliquée. Cela montre l'adéquation directe de GLiNER à la tâche de NER sans nécessiter de fine-tuning pour un domaine aussi précis. Mais aussi met la lumière sur certain limites du cette modéle .

9 Limites et perspectives

L'analyse révèle plusieurs limites significatives du modèle GLiNER dans le contexte biomédical. Le modèle présente des difficultés de segmentation, notamment avec les entités imbriquées où la reconnaissance d'une entité longue masque ses sous-composants, et peine à traiter les composés techniques complexes. Également , Le modèle confond les rôles des termes médicaux selon leur contexte et montre une sensibilité inappropriée à la casse typographique. Les confusions entre catégories proches (FINDING/DISO) et la sur-généralisation de certains labels (INJURY POISONING) génèrent de nombreuses erreurs bidirectionnelles. Enfin, le modèle souffre d'une couverture lexicale inégale, avec une sous-représentation du vocabulaire subjectif (qualité de vie, évaluations) et une performance très variable selon les synonymes utilisés. Ces limitations soulignent le besoin d'optimisation des listes de synonymes, d'amélioration des définitions

de labels et de stratégies de post-traitement pour maximiser le potentiel de GLiNER en extraction d'informations biomédicales.

Pour surmonter ces limites et étendre les capacités des modèles de NER, plusieurs pistes de recherche et de développement peuvent être explorées :

— **Apprentissage par renforcement avec feedback humain** : Intégrer un mécanisme de feedback humain dans la boucle d'apprentissage pour un modèle de type GLiNER pourrait permettre au modèle de s'adapter plus rapidement aux nuances du domaine. Les experts du domaine pourraient valider ou corriger les prédictions, et ce feedback serait utilisé pour affiner les poids du modèle ou ajuster la sélection des synonymes dynamiquement.

— **Développement de techniques de simplicité contextuelle avancées** : Il serait crucial de doter les modèles de mécanismes plus sophistiqués pour simplifier les termes médicaux en fonction de leur rôle et de leur contexte spécifique au sein d'une phrase. Cela pourrait impliquer l'utilisation de modèles de connaissances profondes capables de raisonner sur les relations sémantiques.

— **Gestion améliorée des entités imbriquées et multi-tokens** : Des architectures de modèles ou des stratégies de post-traitement spécifiques pourraient être développées pour mieux gérer les entités composées et imbriquées, en s'assurant que tous les composants pertinents sont identifiés et correctement classifiés.

— **Intégration d'un LLM avec une base de connaissances** : Coupler un grand modèle de langage (LLM) avec une base de connaissances biomédicale structurée (ontologies, graphes de connaissances) pourrait considérablement réduire les "hallucinations" (informations inventées ou incorrectes) et renforcer la détection fine des entités. Le LLM utiliserait la base de connaissances pour valider, affiner et contextualiser ses prédictions, augmentant ainsi la précision et la fiabilité.

En abordant ces limites par des approches innovantes, il sera possible de faire progresser significativement la reconnaissance d'entités nommées dans des domaines complexes comme le biomédical, rendant ces outils encore plus précieux pour la recherche et les applications cliniques.

Références

[1] Arindam Ashok et al. "Disentangled Attention for Personalized Recommendation". In : *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2023), p. 2733-2737.

[2] Yiming Cui et al. “Revisiting Pre-trained Language Models for Named Entity Recognition”. In : *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), p. 6649-6659.

[3] *Disentangled Attention Mechanism*. Consulté le 10 juillet 2025. URL : https://paperswithcode.com/method/disentangled-attention-mechanism (visité le 10/07/2025).

[4] *Équipe SIG (Sémantique et Ingénierie des Connaissances)*. Consulté le 10 juillet 2025. IRIT. URL : https://www.irit.fr/equipe/sig/ (visité le 10/07/2025).

[5] Cyril Grouin et al. “NEREL-BIO : A Richly Annotated Corpus for Named Entity Recognition in French Clinical Texts”. In : *Bioinformatics* 39 (4 2023), btad161. DOI : 10.1093/bioinformatics/btad161. URL : https://academic.oup.com/bioinformatics/article/39/4/btad161/7099619.

[6] LLC Kanerika. “Named Entity Recognition : A Comprehensive Guide to NLP’s Key Technology”. In : *Medium* (2023). Consulté le 10 juillet 2025. URL : https://medium.com/%40kanerika/named-entity-recognition-a-comprehensive-guide-to-nlps-key-technology-636a124eaa46 (visité le 10/07/2025).

[7] Jing Li et al. “A Survey on Deep Learning for Named Entity Recognition”. In : *arXiv preprint arXiv :2411.05057* (nov. 2024). Consulté le 10 juillet 2025. URL : https://arxiv.org/abs/2411.05057.

[8] Xiao Liu et al. “Deep learning for named entity recognition : a review”. In : *Neural Computing and Applications* (2024). URL : https://link.springer.com/article/10.1007/s00521-024-09532-1.

[9] Amit Nadgeri et al. “A survey on Named Entity Recognition and classification”. In : *Computers in Industry* 100 (2018), p. 306-322. ISSN : 0166-3615. DOI : https://doi.org/10.1016/j.compind.2018.06.001. URL : https://www.sciencedirect.com/science/article/pii/S0010482518302105.

[10] *Survey on Named Entity Recognition : From Traditional Approaches to Deep Learning Models*. Consulté le 10 juillet 2025. URL : https://arxiv.org/html/2411.05057v1#:~:text=Ontology%20Population%2C%20and%20so%20on,the%20developing%20unsupervised%20learning%20methods (visité le 10/07/2025).

[11] Pengfei Wang et al. “InstructNER : A Unified Instruction-Tuning Framework for Named Entity Recognition”. In : *arXiv preprint arXiv :2402.00000* (2024).

[12] Maminiaina ZARATIANA et al. “Named Entity Recognition with Transformers in Malagasy : A Case Study”. In : *Proceedings of the 2nd Workshop on Deep Learning for Low-Resource Languages* (2023), p. 65-74.

[13] *Zero-Shot Named Entity Recognition with Large Language Models : A Comprehensive Study*. Consulté le 10 juillet 2025. URL : https://arxiv.org/html/2411.08868v1 (visité le 10/07/2025).

[14] Mengqi ZHOU et al. “Leveraging External Knowledge for Low-Resource Named Entity Recognition”. In : *arXiv preprint arXiv :2308.00000* (2023).