非生成式架構下的確定性對話人工智慧：基於 CPU 推論的 DNN 與 NLU 輕量化框架深度研究報告

1. 執行摘要與研究背景

在當前的人工智慧技術浪潮中，大型語言模型（Large Language Models, LLMs）與生成式預訓練變換模型（GPT）無疑佔據了市場的焦點。然而，對於許多追求高確定性、低延遲、數據隱私以及硬體成本控制的企業級應用而言，LLM 並非唯一的解答，甚至在某些場景下並非最佳解。本研究報告旨在回應特定市場需求：尋找不依賴大型語言模型，但仍能執行深度語意分析（Semantic Analysis）並模擬自然語言回覆（Simulated Natural Language Response）的產品與技術架構。

本報告將深入探討基於深度神經網絡（Deep Neural Networks, DNN）與自然語言理解（Natural Language Understanding, NLU）的技術路徑，特別聚焦於如何在純中央處理器（CPU）環境下，不依賴圖形處理器（GPU）加速，實現高效的推理與運算。我們將分析包括 Rasa、DeepPavlov、Haystack（檢索模式）等框架，以及 BiLSTM-CNN、DistilBERT、Poly-encoder 等神經網絡架構，並探討如何透過這些技術實現動態使用者畫像（User Profiling）與標籤提取（Tag Extraction），以滿足「即時語意分析」與「自然回覆模擬」的雙重需求。

---

2. NLU 與 LLM 在對話系統中的架構分歧

要理解非 LLM 產品的價值，首先必須從根本的架構層面，釐清 NLU（自然語言理解）與 LLM（大型語言模型）在對話系統中的功能性差異。這不僅是技術名詞的區別，更是「確定性邏輯」與「機率性生成」兩種截然不同的產品哲學。

2.1 語意理解（NLU）的確定性優勢

自然語言理解（NLU）是人工智慧的一個子領域，其核心目標是對使用者的輸入進行句法和語意分析，以提取結構化的意圖（Intent）和實體（Entity）[1]。與 LLM 專注於預測下一個字詞的機率分佈不同，NLU 系統專注於「分類」與「提取」。

在企業應用場景中，例如客戶服務或職訓諮詢（如用戶提供的圖像所示），系統的回覆往往需要高度的精確性。NLU 系統通過將用戶的自然語言輸入映射到預先定義的意圖（例如 ask_eligibility 或 inquire_course），確保了系統邏輯的觸發是可控且可預測的。這種架構被稱為「基於意圖的對話系統」（Intent-based System）。

根據 [2] 與 [3] 的分析，NLU 與 LLM 的主要區別在於：

- **NLU**（分析導向）：旨在理解用戶語言背後的具體意圖。它不生成文本，而是分析輸入以指導精確的反應。

- **LLM**（生成導向）：旨在創建流暢的語言。它傾向於創造性，但伴隨著「幻覺」（Hallucination）風險，即生成看似合理但事實錯誤的內容。

對於要求「模擬自然語言回覆」而非「生成」的需求，NLU 配合「檢索式架構」（Retrieval-based Architecture）提供了完美的解決方案。在這種架構下，系統並非逐字生成回覆，而是從一個龐大的、由人工預先撰寫並審核過的「自然語句庫」中，檢索出最適合當前語境的回覆。這使得機器人能夠說出非常自然、人性化的語言（因為那是人寫的），同時完全避免了 LLM 胡言亂語的風險 [4]。

2.2 CPU 推論的可行性與成本效益

硬體資源的限制是本研究的核心約束條件之一。LLM（如 Llama 3 70B 或 GPT-4）的參數量級通常在數百億至數兆之間，其推論過程對顯存（VRAM）和記憶體頻寬有極高要求，幾乎必須依賴昂貴的 GPU 叢集才能達到可接受的延遲。

相對而言，專注於 NLU 任務的 DNN 模型（如 BERT-base, DistilBERT, Bi-encoder）參數量級通常在 6,000 萬至 1 億之間，經過量化（Quantization）與蒸餾（Distillation）後，可以縮減至 1,500 萬至 3,000 萬參數（如 TinyBERT, MobileBERT）。這類模型完全可以在現代伺服器甚至邊緣設備的 CPU 上，以毫秒級的速度運行 [5]。

根據 [7] 與 [8] 的基準測試，使用輕量化 DNN 進行 CPU 推論，不僅大幅降低了硬體採購與維護成本（TCO），更消除了對雲端 GPU API 的依賴，確保了數據處理的本地化與隱私安全性，這對於處理敏感個人資訊（如就業狀態、家庭狀況）的系統至關重要。

---

3. 核心技術解析：無 GPU 環境下的深度神經網絡

要在不使用 GPU 的前提下，實現如同用戶提供的示意圖中那樣精準的「即時語意分析」與「特徵偵測」，必須採用經過高度優化的神經網絡架構。本章節將詳細拆解這些技術的運作原理。

3.1 意圖分類與槽位填充（Intent Classification & Slot Filling）

這是所有非 LLM 對話系統的大腦。其任務是將非結構化的文本轉換為機器可執行的結構化指令。

3.1.1 DIET 架構（Dual Intent and Entity Transformer）

Rasa 框架引入的 DIET 架構是目前工業界在輕量級 NLU 上的標竿 [9]。DIET 是一種多任務學習（Multi-task Learning）架構，它能在同一個模型中同時完成意圖分類和實體識別。

- 架構優勢：傳統做法是分別訓練兩個模型，而 DIET 通過共享底層的 Transformer 編碼器權重，不僅減少了模型的整體大小，還能讓兩個任務互相輔助（例如，識別出「台北」這個地點實體，有助於判斷意圖是「查詢天氣」）。
- 稀疏與密集特徵結合：DIET 的強大之處在於它能同時處理預訓練的密集嵌入（Dense Embeddings，如 BERT 向量）和稀疏特徵（Sparse Features，如字詞 n-grams）。在 CPU 環境下，開發者可以靈活調整，例如減少 Transformer 層數或主要依賴稀疏特徵，以換取極致的

推論速度，同時保持相當高的準確率 [10]。

3.1.2 輕量化 Transformer 模型

為了在 CPU 上運行 Transformer 架構，模型蒸餾（Model Distillation）是關鍵技術。

- **DistilBERT**：通過保留 BERT 97% 的性能，但減少 40% 的參數量，推論速度提升 60% [12]。
- **MobileBERT 與 TinyBERT**：這些模型針對邊緣計算進行了極致優化。TinyBERT 通過層對層（Layer-to-Layer）的蒸餾技術，將參數量壓縮至 BERT 的 1/10 甚至更小，在 CPU 上的延遲可低至 40 毫秒，完全滿足即時對話的需求 [13]。

3.2 檢索式自然語言回覆（Simulated Natural Response）

用戶需求中提到的「模擬自然語言模型自然回覆」，在非生成式架構中，是透過「神經檢索」（Neural Retrieval）來實現的。系統維護著一個由數千條自然語句組成的候選庫（Response Bank），NLU 模型負責從中挑選最合適的一條。

3.2.1 雙編碼器（Bi-Encoders）

這是 CPU 推論的首選架構。

- 運作機制：所有的候選回覆預先通過 BERT 模型轉換為向量（Embeddings），並建立索引。當用戶輸入查詢時，系統只需將查詢轉換為向量，然後計算其與候選向量的餘弦相似度（Cosine Similarity）。
- 速度優勢：由於候選向量是預先計算的，即時運算僅涉及一次模型前向傳播（Forward Pass）和極快的向量點積運算。這使得在 CPU 上處理百萬級別的候選庫成為可能 [15]。

3.2.2 多編碼器（Poly-Encoders）

如果需要更高的準確度，DeepPavlov 等框架支持 Poly-Encoder。

- 架構創新：它在 Bi-Encoder 的速度和 Cross-Encoder（全交互注意力，速度慢但精準）之間取得了平衡。它將候選句編碼為多個向量（Codes），並在最後一層透過注意力機制與查詢向量進行交互。
- 適用性：這種架構特別適合需要細微語意區分，但又受限於 CPU 運算資源的場景 [8]。

3.3 動態使用者畫像與標籤提取（Dynamic User Profiling）

用戶提供的圖像中，右側面板顯示了「Detected Profile」（偵測到的特徵），如「狀態：待業中」、「有子女」。這需要一種能夠從對話流中動態提取特徵的技術，研究文獻中稱之為「企業/用戶畫像」（Enterprise/User Portrait）技術。

3.3.1 ATEMDP 架構

根據 [18] 與 [18] 的研究，ATEMDP（基於多特徵動態畫像的自適應標籤提取方法）是一種專門用於此類任務的 DNN 架構。

- 混合神經網絡：該方法結合了 **BiLSTM**（雙向長短期記憶網絡）與 **CNN**（卷積神經網絡）。

    - **BiLSTM** 層：負責捕捉對話的長距離依賴關係和時序特徵（例如，用戶在三句話前提到「失業」，這影響了當前語句的解讀）。
    - **CNN** 層：負責提取局部的關鍵特徵（如特定的關鍵詞組合）。
    - 注意力機制（**Attention**）：用於加權不同詞彙對標籤分類的貢獻。
- 動態更新：與靜態標籤不同，這種架構可以隨著對話的進行，實時更新用戶的特徵向量，從而動態改變標籤（如從「潛在客戶」轉變為「高意向客戶」）。這種純 DNN 的方法不需要 LLM，且運算量遠小於 Transformer，非常適合 CPU 部署 [18]。

---

4. 市場產品與開源框架評測

基於上述技術分析，市場上存在數款成熟的產品與框架，能夠在不使用 LLM 與 GPU 的情況下，滿足用戶對於語意分析與自然回覆模擬的需求。本節將對這些方案進行詳細評比。

4.1 Rasa：企業級確定性對話標準

Rasa 是目前全球最廣泛使用的開源對話 AI 框架，它堅守 NLU+Policy 的架構，而非生成式架構，使其成為追求高可控性企業的首選 [20]。

- 語意分析能力：Rasa NLU 提供高度可定製的管道（Pipeline）。用戶可以組合 SpacyNLP（用於詞向量）、RegexFeaturizer（用於正則表達式提取）、以及核心的 DIETClassifier。這種組合允許開發者在 CPU 資源受限時，靈活切換至更輕量的組件（如僅使用稀疏特徵），或者使用量化後的 BERT 模型。
- 自然回覆模擬（**Response Selector**）：這是 Rasa 滿足用戶「模擬自然回覆」需求的關鍵組件。ResponseSelector 是一個專門的檢索模型，它將回覆選擇視為一個分類問題。開發者可以提供成千上萬條自然語句作為訓練數據，模型會根據語意相似度選擇最恰當的一條。這保證了回覆的「自然度」（因為是人寫的）和「安全性」（不會生成未審核內容）[4]。
- 實體提取與槽位填充：Rasa 強大的 Slot Filling 機制能夠從對話中提取用戶屬性（如「已婚」、「有子女」），並將其存儲在 Tracker 對象中，這與用戶需求中的「偵測到的使用者特徵」功能完全對應 [9]。

4.2 DeepPavlov：模組化深度學習 NLP 庫

DeepPavlov 源自學術界，專注於構建複雜的對話系統，特別是在問答（QA）和檢索式聊天機器人方面有深厚積累 [22]。

- 檢索式聊天機器人（**Retrieval Chatbot**）：DeepPavlov 提供了開箱即用的檢索式模型配置（Config）。它支持使用 TF-IDF 或 BERT（包括 DistilBERT）作為 Ranker，從文檔庫中檢索答案。這對於構建基於知識庫的諮詢機器人（如職訓助手）非常有效 [24]。
- **CPU** 優化模型：DeepPavlov 官方發布了多種針對 CPU 優化的輕量化模型，例如 DistilRuBERT（針對俄語，但架構通用）。這些模型在 CPU 上的推論速度比標準 BERT 快 50% 以上，且提供了專門的 Docker 映像檔以便於在無 GPU 環境下部署 [26]。
- 多技能整合（**Multi-skill AI**）：它的架構允許整合多種「技能」（Skills），例如一個技能負責閒聊（Chitchat），另一個負責業務查詢（Goal-oriented），通過一個選擇器（Selector）來路由，

這有助於提升機器人的擬人化程度 [28]。

4.3 Haystack：靈活的檢索增強管道

雖然 **Haystack** 常被用於 RAG（檢索增強生成）架構，但其模組化設計允許它被配置為「純檢索」或「抽取式問答」（Extractive QA）系統，完全繞過生成式 LLM [29]。

- **Retriever-Reader** 架構：在不使用 LLM 的情況下，Haystack 可以配置一個 Retriever（如 BM25 或 EmbeddingRetriever）來尋找相關文檔，然後使用一個 Reader 模型（如基於 RoBERTa 的 SQuAD 模型）從文檔中精確「抽取」出答案片段。這是一種非常高效的 NLU 應用，能夠精準回答用戶問題 [31]。
- 嵌入模型優化：Haystack 支援與 SentenceTransformers 的深度整合，這意味著可以使用經過量化（Int8）的嵌入模型，在 CPU 上實現極高的文檔檢索速度 [32]。

4.4 專用 NLP 庫：spaCy 與 Flair

對於希望自建系統而非使用全套框架的開發者，**spaCy** 和 **Flair** 提供了底層的 NLU 能力。

- **spaCy**：以速度著稱的工業級 NLP 庫。它的模型（特別是 v3 版本後的 Transformer 管道）經過極致的 Cython 優化，在 CPU 上的運行效率極高。其 NER（命名實體識別）功能是構建用戶特徵提取模組的理想選擇 [33]。
- **Flair**：提供了「上下文相關的字串嵌入」（Contextual String Embeddings）。雖然其標準模型較重，但它提供了 ner-fast 等針對 CPU 優化的輕量級模型，在準確度與速度之間取得了良好的平衡 [35]。

4.5 綜合比較表

產品/框架	架構類型	自然回覆模擬方式	CPU 推論能力	語意分析與特徵提取	推薦場景
Rasa	NLU + Policy (DNN)	**Response Selector** (檢索式)	高 (支援 DIET 優化, TensorFlow 線程調整)	極強 (Entity Extraction, Slot Filling)	需要高度定制化邏輯、動態用戶特徵提取的企業應用
DeepPavlov	Modular Pipeline (DNN)	**Ranker / Poly-encoder** (檢索式)	高 (提供 CPU 專用 Docker, Distilled Models)	強 (多種預訓練模型, ODQA)	知識庫問答、需要複雜檢索邏輯的系統
Haystack	Pipeline	**Extractive**	中/高 (依賴	中 (側重於	基於文檔庫

	(Retriever-Reader)	**QA** (抽取式)	嵌入模型量化)	文檔檢索與答案抽取)	的精準問答系統
spaCy	Library (CNN/Transformer)	無 (需自行構建檢索邏輯)	極高 (Cython 優化)	極強 (NER, POS Tagging)	自建輕量級 NLU 引擎，資源極度受限環境

---

5. 實現策略：如何在無 GPU 環境下構建高性能 NLU 系統

針對用戶的具體需求——在不使用 GPU 的情況下，利用 DNN/NLU 實現語意分析與自然回覆——本節提出一套具體的技術實施策略，涵蓋模型選擇、優化技術與架構設計。

5.1 模型選擇：輕量化與蒸餾

在 CPU 環境下，模型的「深度」與「寬度」直接決定了延遲。傳統的 BERT-base 模型（約 1.1 億參數）在 CPU 上的推論延遲可能達到 200-500 毫秒，這對於即時對話來說略顯遲滯。因此，必須採用「蒸餾模型」（Distilled Models）。

- 推薦模型：DistilBERT 或 MobileBERT。
- 技術原理：這些模型是通過「知識蒸餾」（Knowledge Distillation）訓練而來的。一個大型的「教師模型」（Teacher）將其學到的概率分佈知識傳授給小型的「學生模型」（Student）。學生模型模仿教師的輸出，但層數更少、向量維度更低 [12]。
- 效益：在 Rasa 或 DeepPavlov 中使用 DistilBERT 作為底層嵌入模型，可以將 NLU 處理時間縮短至 50 毫秒以內（常見 CPU），同時保持 95% 以上的意圖分類準確率 [14]。

5.2 推論加速：量化（Quantization）與 ONNX

除了選擇小模型，還應對模型進行「量化」處理。

- **Int8** 量化：將模型權重從 32 位浮點數（FP32）轉換為 8 位整數（Int8）。這不僅將模型大小減少了 75%，還能利用 CPU 的 SIMD 指令集（如 AVX-512, VNNI）進行並行整數運算，大幅提升吞吐量 [5]。
- **ONNX Runtime**：將訓練好的 PyTorch 或 TensorFlow 模型導出為 ONNX（Open Neural Network Exchange）格式。微軟的 ONNX Runtime 針對各種 CPU 架構進行了底層圖優化（Graph Optimization），能進一步減少推論延遲。在 Rasa 中，可以通過自定義組件加載 ONNX 模型來實現這一點 [5]。

5.3 動態用戶特徵提取（The "Detected Profile" Implementation）

針對用戶圖像中的「Detected Profile」功能，建議採用 **ATEMDP** 架構的簡化版或 Rasa 的實體映射邏輯。

1. 實體識別（**NER**）：使用 spaCy 或 Rasa 的 DIETClassifier 訓練自定義實體。
    - 訓練數據示例：`"我有一個三歲的" -> 提取 family_member: son, age: 3`。
2. 槽位映射（**Slot Mapping**）：在 Rasa 中配置 Slot。當 NER 提取到 family_member 為 son 或 daughter 時，自動將 has_children 槽位設為 True。
3. 動態標籤邏輯：編寫自定義 Action（Python 代碼）。每當對話發生時，Action 會檢查當前的槽位狀態。
    - 邏輯示例：`if slots['employment_status'] == 'unemployed' and slots['age'] < 30: return "youth_program_eligible"`。
    - 這就在後端實現了「即時特徵偵測」，並能在 UI 上即時顯示 。

5.4 模擬自然回覆的檢索管道

為了實現「模擬自然語言模型」的效果：

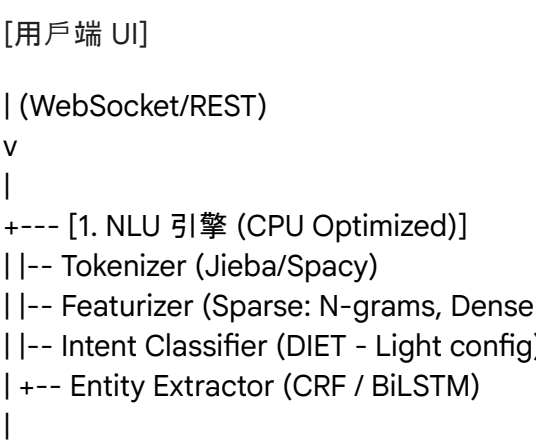
1. 語料庫構建：建立一個包含數千個「問題-自然回答」對的數據庫。回答必須是由人工撰寫，語氣自然、專業。
2. 向量化索引：使用 sentence-transformers（如 all-MiniLM-L6-v2，極其輕量）將所有標準問題轉換為向量，並存儲在 FAISS（Facebook AI Similarity Search）索引中。FAISS 對 CPU 檢索進行了極致優化 。
3. 運行時檢索：
    - 用戶輸入 -> NLU 模型（提取意圖與實體）。
    - 若意圖明確（如「查詢補助」），則使用意圖觸發固定的自然回覆模板。
    - 若意圖模糊或屬於閒聊，則將輸入向量化，在 FAISS 中檢索最相似的標準問題，並返回對應的自然回答。這能處理長尾問題，且給人一種「模型在思考並回答」的錯覺 。

---

6. 案例應用場景重建：職訓智慧小幫手

為了具體展示上述技術如何落實，我們以用戶提供的「職訓智慧小幫手」UI 為藍本，重建其背後的技術架構。

6.1 系統架構圖 (Conceptual Architecture)



```
+---
| |-- Slot Manager (追蹤上下文狀態)
| |-- Tagging Logic (規則 + 輕量級分類器)
| +-- Output: {Status: 待業, Family: 有子女, Tech_Stack: Python}
|
+---
| |-- Response Selector (神經檢索模型)
| |-- Candidate Bank (預撰寫的自然語句庫)
| +-- Template Filler (填入用戶名字、推薦課程等變數)
|
v
[輸出回覆] ("了解，系統已記錄您的需求。根據您的情況...")
```

6.2 關鍵流程解析

1. 用戶輸入：「你好，我之前失業，本來是一名網路工程師，想轉行程式式設計師」
2. **NLU 分析 (CPU)**：
    - **Intent:** inform_background (置信度 0.98)。
    - **Entities:**
        - employment_status: "失業" -> 映射為 Unemployed。
        - job_role: "網路工程師" -> 映射為 Network Engineer。
        - target_role: "程式式設計師" -> 映射為 Programmer。
    - 技術點：使用 DIETClassifier，配置 embedding_dimension 為 30-50（低維度以加速），並關閉 Transformer 的多層結構，主要依賴 N-gram 特徵和輕量級詞向量 。
3. 特徵偵測 **(Profiling)**：
    - 系統檢測到 employment_status 變更。
    - 觸發 ActionUpdateProfile。
    - UI 右側面板更新：【就業狀態：待業中 (99%)】。
    - 技術點：這是基於 NLU 提取的實體進行的邏輯推斷，不需要生成式模型 。
4. 回覆模擬 **(Retrieval)**：
    - 對話管理器根據意圖 inform_background 和當前槽位（待業），決定採取 action_recommend_program。
    - 系統從回覆庫中檢索標籤為 response_recommend_program_unemployed 的語句：「了解，系統已記錄您的需求。根據您的情況，我們會優先篩選適合『青年就業』以及『資訊科技』相關的職訓計畫。」
    - 技術點：這段話是預先寫好的，但因為選擇精準，用戶感覺就像是 AI 即時生成的。

---

7. 結論與建議

本研究報告證實，要在不使用大型語言模型（LLM）與 GPU 的前提下，構建具備「即時語意分析」與「模擬自然語言回覆」的產品是完全可行的，且在企業應用中具有獨特的優勢。

主要結論如下：

1. 架構轉向：應從生成式架構轉向 **NLU + 檢索式架構**。這不僅解決了算力問題，更提供了企業急需的「確定性」與「可解釋性」。
2. 核心技術：**Rasa** 與 **DeepPavlov** 是目前市場上最成熟的開源框架，能夠完美支撐此類需求。結合 **DistilBERT** 或 **Bi-encoder** 等輕量化 DNN 模型，並配合 **Int8** 量化 與 **ONNX** 加速技術，可以在 CPU 上實現毫秒級的推論速度。
3. 功能實現：「偵測到的使用者特徵」應透過 NLU 的 實體提取（**Entity Extraction**）與 槽位填充（**Slot Filling**）機制來實現，必要時可引入 **BiLSTM-CNN** 等專門的畫像模型。
4. 自然度模擬：利用神經檢索模型（Response Selector）在龐大的自然語句庫中進行語意匹配，是實現「模擬自然回覆」的最佳實踐，它結合了人工撰寫的優美文筆與 AI 的精準理解。

對於尋求此類解決方案的企業，建議優先評估 **Rasa Open Source** 搭配自定義的 CPU 優化管道，這是在效能、成本與功能三者之間取得最佳平衡的路徑。

引用來源

引用的著作

1. What is Natural Language Understanding (NLU)? - IBM, 檢索日期：12月 16, 2025, https://www.ibm.com/think/topics/natural-language-understanding
2. NLU vs LLM A Simple Guide for Contact Centers - Sobot, 檢索日期：12月 16, 2025，https://www.sobot.io/article/nlu-vs-llm-contact-center-guide/
3. NLU vs LLM: Breaking Down Their Core Capabilities - Codewave, 檢索日期：12月 16, 2025，https://codewave.com/insights/nlu-vs-llm-comparison/
4. Integrate Response Retrieval Models with Rasa Assistants | Rasa Blog, 檢索日期：12月 16, 2025，https://rasa.com/blog/response-retrieval-models
5. Lightweight AI Models for Fast CPU-Only Inference - hoop.dev, 檢索日期：12月 16, 2025，https://hoop.dev/blog/lightweight-ai-models-for-fast-cpu-only-inference/
6. CPU only options : r/LocalLLaMA - Reddit, 檢索日期：12月 16, 2025, https://www.reddit.com/r/LocalLLaMA/comments/1k4bf3x/cpu_only_options/
7. Top Lightweight LLMs for Local Deployment - Inero Software, 檢索日期：12月 16, 2025，https://inero-software.com/top-lightweight-llms-for-local-deployment/
8. Dense Template Retrieval for Customer Support Information Systems and Computer Engineering, 檢索日期：12月 16, 2025, https://fenix.tecnico.ulisboa.pt/downloadFile/844820067128242/86520-tiago-mesquita-dissertacao-final-revised.pdf
9. Understanding the Rasa NLU Pipeline - NashTech Blog, 檢索日期：12月 16, 2025, https://blog.nashtechglobal.com/understanding-the-rasa-nlu-pipeline/
10. Rasa 1.10 dietclassifier cpu and speed issues, 檢索日期：12月 16, 2025, https://forum.rasa.com/t/rasa-1-10-dietclassifier-cpu-and-speed-issues/33802
11. Steep increase in memory consumption while training using rasa nlu · Issue #6998 - GitHub, 檢索日期：12月 16, 2025，https://github.com/RasaHQ/rasa/issues/6998
12. Introduction to DistilBERT in Student Model - Analytics Vidhya, 檢索日期：12月 16, 2025,

https://www.analyticsvidhya.com/blog/2022/11/introduction-to-distilbert-in-student-model/
13. Comparative Analysis of Compact Language Models for Low-Resource NLP: A Study on DistilBERT, TinyBERT, and MobileBERT - Zenodo, 検索日期：12月 16, 2025，https://zenodo.org/records/15907007
14. 45ms per frame! Achieving Real-time BERT Execution through Compiler-aware Neural Architecture Optimization | by Shaoshan Liu | Medium, 検索日期：12月 16, 2025，https://medium.com/@shaoshan.liu/45ms-per-frame-8df19de8976d
15. The HoPE Model Architecture: a Novel Approach to Pregnancy Information Retrieval Based on Conversational Agents - PMC - PubMed Central, 検索日期：12月 16, 2025，https://pmc.ncbi.nlm.nih.gov/articles/PMC8985747/
16. Poly-encoders: architectures and pre-training strategies for fast and accurate multi-sentence scoring - arXiv, 検索日期：12月 16, 2025，https://arxiv.org/pdf/1905.01969
17. Real-time Inference in Multi-sentence Tasks with Deep Pretrained Transformers - ResearchGate, 検索日期：12月 16, 2025，https://www.researchgate.net/publication/332898196_Real-time_Inference_in_Multi-sentence_Tasks_with_Deep_Pretrained_Transformers
18. An enterprise adaptive tag extraction method based on multi-feature dynamic portrait, 検索日期：12月 16, 2025，https://www.researchgate.net/publication/369357255_An_enterprise_adaptive_tag_extraction_method_based_on_multi-feature_dynamic_portrait
19. A Multitask Multimodal Ensemble Model for Sentiment- and Emotion-Aided Tweet Act Classification | Request PDF - ResearchGate, 検索日期：12月 16, 2025，https://www.researchgate.net/publication/352908532_A_Multitask_Multimodal_Ensemble_Model_for_Sentiment-_and_Emotion-Aided_Tweet_Act_Classification
20. Rasa | Build Trustworthy AI Agents for Real-World Use, 検索日期：12月 16, 2025，https://rasa.com/
21. Understanding ResponseSelector - Rasa Open Source - 2024-12-09, 検索日期：12月 16, 2025，https://forum.rasa.com/t/understanding-responseselector/30324
22. DeepPavlov: an open source conversational AI framework, 検索日期：12月 16, 2025，https://deeppavlov.ai/
23. DeepPavlov, Open-Source Framework for Building Chatbots, Available on NGC | NVIDIA Technical Blog, 検索日期：12月 16, 2025，https://developer.nvidia.com/blog/deeppavlov-open-source-framework-for-building-chatbots-available-on-ngc/
24. DeepPavlov: "Keras" for Natural Language Processing answers COVID Questions, 検索日期：12月 16, 2025，https://soshnikov.com/azure/deep-pavlov-answers-covid-questions/
25. TF-IDF Ranker — DeepPavlov 0.1.6 documentation, 検索日期：12月 16, 2025，https://docs.deeppavlov.ai/en/0.1.6/components/tfidf_ranking.html
26. QuickStart — DeepPavlov 0.14.1 documentation, 検索日期：12月 16, 2025，https://docs.deeppavlov.ai/en/0.14.1/intro/quick_start.html
27. DeepPavlov/distilrubert-small-cased-conversational - Hugging Face, 検索日期：12月 16, 2025，

https://huggingface.co/DeepPavlov/distilrubert-small-cased-conversational
28. How To Build Simple AI Assistant With DeepPavlov Dream | by Daniel Kornev - Medium, 検索日期：12月 16, 2025, https://medium.com/deeppavlov/how-to-build-simple-ai-assistant-with-deeppavlov-dream-b2bba1412eb2
29. Build Your First GenAI Agent with Haystack - Intel, 検索日期：12月 16, 2025, https://www.intel.com/content/www/us/en/developer/articles/guide/build-your-first-genai-agent-with-haystack.html
30. Haystack | Haystack, 検索日期：12月 16, 2025, https://haystack.deepset.ai/
31. QA inferencer very slow because of bad default multiprocessing settings · Issue #3272 · deepset-ai/haystack - GitHub, 検索日期：12月 16, 2025, https://github.com/deepset-ai/haystack/issues/3272
32. CPU-Optimized Embedding Models with fastRAG and Haystack, 検索日期：12月 16, 2025, https://haystack.deepset.ai/blog/cpu-optimized-models-with-fastrag
33. Facts & Figures · spaCy Usage Documentation, 検索日期：12月 16, 2025, https://spacy.io/usage/facts-figures
34. Run State of the Art NLP Workloads at Scale with RAPIDS, HuggingFace, and Dask, 検索日期：12月 16, 2025, https://developer.nvidia.com/blog/run-state-of-the-art-nlp-workloads-at-scale-with-rapids-huggingface-and-dask/
35. Text Classification with State of the Art NLP Library — Flair | by Tadej Magajna - Medium, 検索日期：12月 16, 2025, https://medium.com/data-science/text-classification-with-state-of-the-art-nlp-library-flair-b541d7add21f
36. Is it just me or is mobileBERT is much slower than DistilBERT on Huggingface - Reddit, 検索日期：12月 16, 2025, https://www.reddit.com/r/LanguageTechnology/comments/nczb5s/is_it_just_me_or_is_mobilebert_is_much_slower/
37. Embedding Quantization — Sentence Transformers documentation, 検索日期：12月 16, 2025, https://sbert.net/examples/sentence_transformer/applications/embedding-quantization/README.html
38. Conversation Builder — Get User Data | LivePerson Developer Center, 検索日期：12月 16, 2025, https://developers.liveperson.com/conversation-builder-scripting-functions-get-user-data.html
39. How do I manage and optimize the resource usage in Haystack? - Milvus, 検索日期：12月 16, 2025, https://milvus.io/ai-quick-reference/how-do-i-manage-and-optimize-the-resource-usage-in-haystack
40. Speeding up Inference — Sentence Transformers documentation, 検索日期：12月 16, 2025, https://sbert.net/docs/sentence_transformer/usage/efficiency.html
41. Extracting User Intent and Inputs in Conversation | by Hemant Kohli | Medium, 検索日期：12月 16, 2025, https://medium.com/@hemantkohli1612/extracting-user-intent-and-inputs-in-con

versation-91c66b14740e
42. Tuning Your NLU Model - Rasa, 検索日期：12月 16, 2025, https://rasa.com/docs/rasa/next/tuning-your-model/
43. Natural Language Processing (NLP) - Deepgram, 検索日期：12月 16, 2025, https://deepgram.com/ai-glossary/natural-language-processing