Named entity recognition for IT jobs description in Russian language

Лазуков Михаил, Лихота Константин, Одинец Никита

May 2022

Аннотация

В этом проекте решалась задача извлечения именнованных сущностей из описания IT вакансий. В данной работе описан весь процесс решения задачи от сбора данных, до обучения моделей. Ссылка на репозиторий с проектом `https://github.com/mikhaillazukov/ner_it_job_rus`

1 Введение

В наше время информационные технологии прочно вошли в нашу жизнь, а вместе с ними возросла и потребность в эффективном поиске и анализе данных. Каждый день через людей проходит огромное количество информации и важно уметь извлекать из них наиболее важную часть. В данной работе мы решили сконцентрироваться на конкретной задаче, а именно извлечение именнованных сущностей (от англ. Named entity recognition). В качестве области применения выбрали вакансии IT специальностей. Тексты вакансий не на специализированных платформах зачастую не имеют структуры. Вследствие чего, приходится тратить время на поиск нужной информации. Почему же были выбраны именно IT вакансии? Во-первых в IT имеется большое количество профессий, во-вторых казалось бы одинаковые должности могут требовать совершенно разных навыков. Например, должность Junior Backend Developer может требовать от человека знания Java, Spring, Docker, gitlab CI/CD. Из списка требования мы понимаем, что речь идет от разработке серверной части Web приложений и сервисов. А на ту же самую вакансию может иметь совершенно другой смысл, если в качетсве требований будут указаны C++, Linux, Qt5 и т.д. Здесь речи уже идет о разработчике приложений для персональных компьютеров с графическим интерфейсом. Зачастую можно включать необходимую информацию в название вакансии, чтобы соискателем должности было проще подбирать вакансии, но к

сожалению не все HR-специалисты компетентны в вопросах грамотного составления вакансий.

Целью нашего проекта является разработка системы, способной автоматически обрабатывать текстовые данные, содержащие вакансии, и выделять в них важные именованные сущности. Это позволит улучшить процесс анализа вакансий, значительно сократить время поиска и сопоставления информации соискателей. Самым простым примером использования нашего решения может является применение его для составления выжимки информации из вакансии перед публикацией в тематическом канале Telegram. Иным примером может послужить автоматическая проверка публикуемых вакансий. Например, некоторые администраторы каналов имеют четкие критерии к публикациям, а именно наличие информации о заработной плате, требуемые навыки и т.д. Таким образом, в случае если после работы нашего алгоритма не было найдено какой-то сущности, то вакансия в автоматическом режиме может отправляться на доработку.

Проведя исследование в интернете, мы не нашли готовых решения и наборов данных для NER из описаний IT вакансий на русском языке. В результате чего, нами было принято решения о самостоятельном сборе даных и создания решения на их основе.

1.1 Команда

Михаил Лазуков: Сбор сырых данных, обеспечения работы сервера разметки, разметка данных и написание кода для обучение моделей.

Лихота Кностантин: Исследование известных подходов и доступных результатов,разметка данных, написание кода для конвертации разметки, оформление итоговой работы.

Никита Одинец: Обработка сырых данных, разметка данных, эксперименты по обучению моделей, оформление репозитория.

2 Сопутствующие работы

Как уже было сказано в введении, в интеренете отсутствуют наборы данных и готовые решения, которые решают задачу NER для описания вакансий на русском языке. Основная работа в данном проекте заключается в сборе и разметке набора данных, т.к. это наиболее важная часть для решения задачи NER. При наличии хорошо подготовленного набора данных задача дожна довльно успешно с примением современных арихтектур LLM.

В ходе исследования публикаций по теме Named Entity Recognition, была найдена интересная работа[Nyqvist, 2021], которая решает схожую с нашей задачу о поиске именованных сущностей в описаниях вакансий.

Эта статья интересна с точки зрения реализации модели не на англоязычных данных. Из этой статьи мы смогли вынести общий подход для решения нашей задачи, рассмотрели метрики, которые могут использоваться в последующей работе и в целом разобраться с пайплайном решения задачи NER.

Также нас заинтересовала модель Few-Shot NER[Ding et al., 2021], о которой мы узнали из статьи на Хабр. Авторы статьи подняли важную проблему об объеме размеченных данных для современных моделей NER. Размеченных данных на региональных языках очень мало, с обирать их сложно, долго и дорого, особенно в специфических доменах (юридическом, медицинском и других). Для решения этой проблемы предлогается использовать Few-Shot подходов к обучению. "Few-Shot Learning — это задача машинного обучения, в которой модель надо преднастроить на тренировочном датасете так, чтобы она хорошо обучалась на ограниченном количестве новых размеченных примеров. То есть даже на новых классах или доменах данных мы сможем получить качественные предсказания."Данный подход реалищовать не удалось, но в целом его можно будет использовать для дальнейших исследований.

3 Описание модели

В качестве основной модели мы использовали BERT[Devlin et al., 2019] и сравнивали ее с несколькими модификациями. BERT - это модель глубокого обучения, которая представляет собой основополагающую архитектуру в области обработки естественного языка, представленный в 2018 году командой исследователей от Google AI Language.

Архитектура BERT основана на трансформерах, которые являются мощным инструментом для моделирования последовательностей. Трансформеры [Vaswani et al., 2017] позволяют моделировать долгосрочные зависимости в текстовых данных, а BERT использует эту архитектуру для представления слов в контексте их окружения.

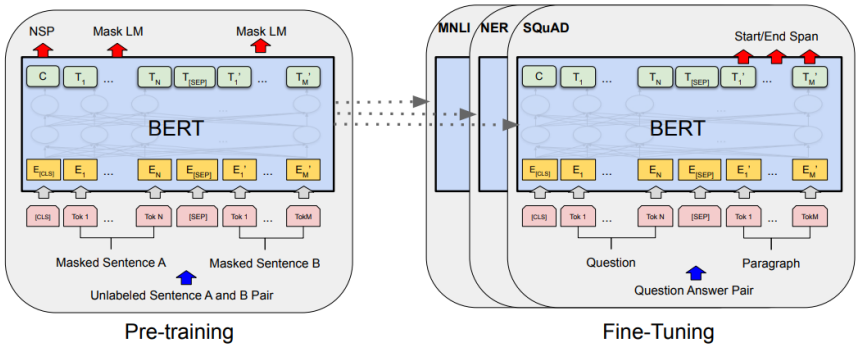

Рис. 1: Архитектура BERT

Основная идея BERT заключается в том, чтобы предобучить модель на большом корпусе текстовых данных. Предварительное обучение модели на большом объеме данных позволяет модели улавливать семантические и синтаксические особенности языка и обладать общими знаниями о тексте.

BERT использует двунаправленную модель, что означает, что она способна анализировать контекст как слева направо, так и справа налево. Это позволяет модели учитывать информацию из всего контекста, что является важным для правильного понимания значения слова или фразы, что очень важно для задачи NER.

Для представления текстовых данных BERT использует WordPiece эмбеддинги, которые разбивают слова на подслова. Это помогает справиться с проблемой неизвестных слов и позволяет модели обобщать на основе семантических и лексических связей между словами.

Однако, несмотря на свои преимущества, BERT также имеет некоторые ограничения. Он требует больших вычислительных ресурсов и времени для обучения и инференса, а также больших объемов размеченных данных для эффективного дообучения на конкретной задаче NER. Для решения этой проблемы был разработан DistilBERT, который удаляет некоторые слои и уменьшает размерность, чтобы сделать модель компактной и более вычислительно эффективно. В качестве второй модели был выбра ruBert от DeepPavlov. Ее главное преимущество в том что она изначально была обучена на русском языке.

4 Набор данных

При поиске открытых датасетов для нашего проекта по извлечению именованных сущностей из вакансий мы обнаружили, что такие данные не были доступны. В связи с этим, мы приняли решение собрать свой собственный датасет. Для этого мы провели парсинг нескольких Telegram-каналов, которые регулярно публикуют вакансии в этой

области. В результате чего нами было собрано порядка 90000 сообщений, которые необходимо было обработать. При просмотре данных было замечено, что некоторые из них были написаны на английском. С помощью регулярных выражений были отброшены те вакансии, в которых содержался только английский текст. После чего было замечено, что часть вакансий не относятся к IT индустрии. Для удаления таких вакасий был составлен список слов, при наличие которых, вакансия удалялась, например: школа, художник, трек, маркетолог и т.п.. Также в некоторых каналах позволялось общаться пользователям. Такой кластер ложных данных был отделен по количество символов. Порог был выбран эмприческим путем и равен 210.

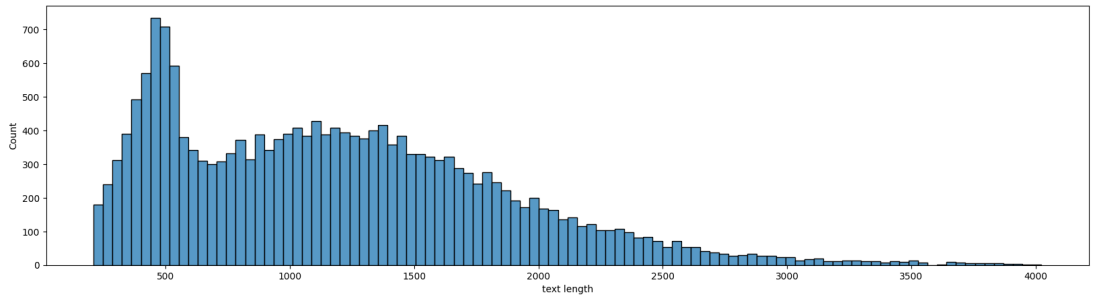

Рис. 2: Распределение описаний вакансий по количеству символов

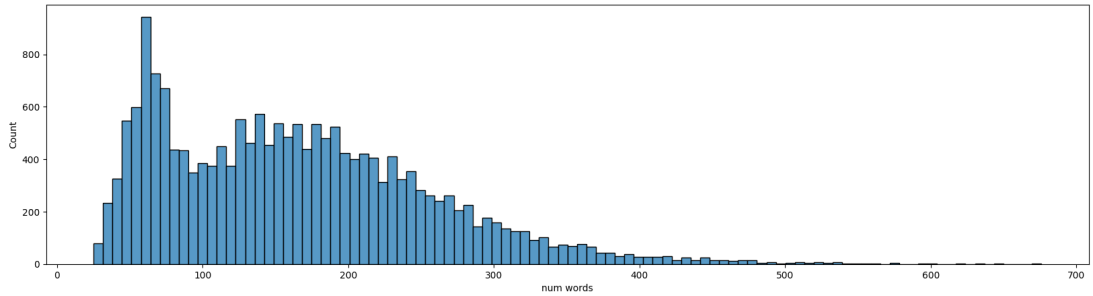

Рис. 3: Распределение описаний вакансий по количеству слов

Далее необходимо было подготовить сами данные. В них присутсвует обилие смайликов, непонятные символы, в одном из Telegram-каналов присутствовала шаблонная вставка текста. Также необходимо было избавиться от излишней табуляции, хэштегов и ссылок. После удаления неактуальных записей нам удалось собрать около 18 000 вакансий.

Рис. 4: Wordcloud для полученных данных

Из облака мы видим, что довольно много слов присущих для вакансий. Например, опыт, разработки, знание, занятость и т.д.

На следующем этапе мы приступили к разметке датасета. Мы выделили 3 типа именованных сущностей из каждой вакансии, которые по мнению авторов являются наиболее информативными: название вакансии, навыки, и зарплата. В качестве типа разметки было выбрано BIO.

Далее мы приступили к выборы средства разметки. Нами было проанализировано несколько современных инструментов: lighttag.io, prodi.gy, awesome-data-labelling, Label-studio. Среди перечисленных победителем вышел Label-studio, т.к. он open-source, бесплатный для использования и его можно легко развернуть на сервере. Приложение для разметки было установлено на учебный сервер, чтобы у всех участников проекта была возможность принять участие в проекте.

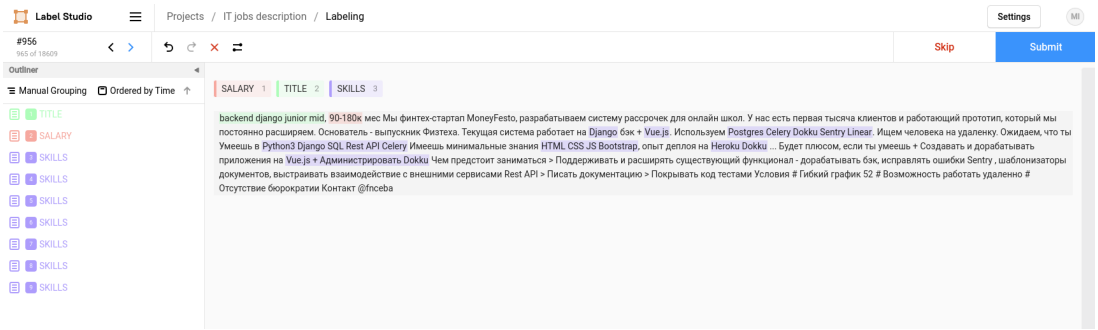

Рис. 5: Иллюстрация процесса разметки

Разметку выкачивали в формате Conll. После чего переводили его в формат json, для удобного считывания при обучении.

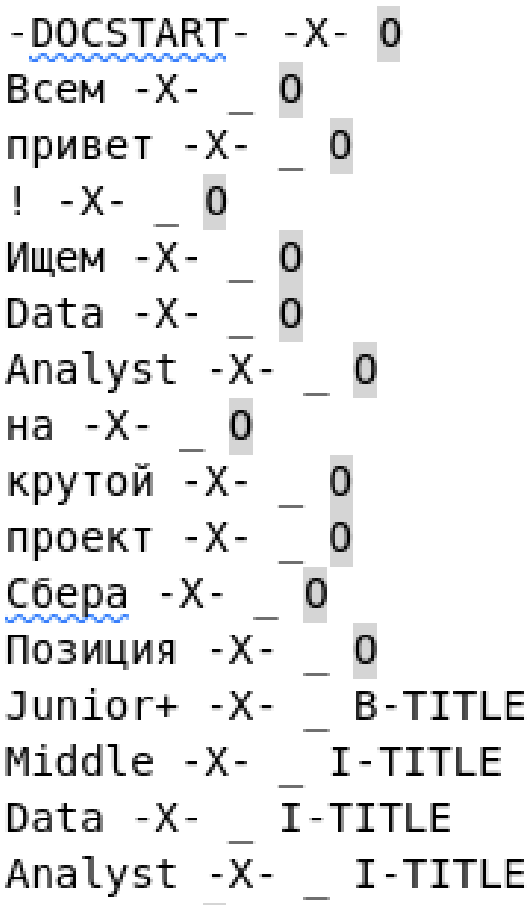

Рис. 6: Разметра в формате Conll

Скорость разметки была не велика, данный процесс довольно трудоемкий и было принято решение попытаться его ускорить.

Первый способ - разметка с помощью языковых моделей (Large Language Models). Были попробованы open-source модели, но к сожалению, они не имели довольно большого качества. Далее было принято решение попробовать GPT-3.5, GPT-4, через доступные API. Эти модели размечали сущности гораздо лучше. К сожалению, мы столкулись с нескольними проблемами при использовании этого метода. Во-первых, Большинство доступных моделей имели ограничение на количество запросов, поэтому нам пришлось создавать множество аккаунтов для работы с ними. Следующая проблема возникла при обработке результатов. Мы столкнулись с проблемой неоднородных результатов. Модели часто изменяли окончания слов, выводили лишнюю информацию или переводили слова на другой язык. В связи с этим от этого подхода пришлось отказаться.

Второй способ - разметка с использованием регулярных выражений и словаря. Как правило, зарплату можно выделить по регулярному выражению, где на первом месте будет идти слово "вилка"или "зарплата". Навыки можно выделить с помощью словаря. Например, Python, Java, SQL и т.д. Для названий вакансий можно определить такие

слова, как Middle, Junior, разработчик и т.п. Этот метод позволил нам определить некоторые шаблоны и правила для извлечения именованных сущностей из текста вакансий. Данный метод отработал лучше, чем предыдущей, но он также оставляет за собой большое количество пропусков, или ошибки в выделении (Например, Python в имени вакансии Python Developer отмечается как часть сушности SKILLS, вместо TITLE). Многие проблемы можно исправив написание более строгих шаблонов, но в таком случае будет выделяться потенциально меньше слов.

Таким образом, для разметки использовался подход с предразметкой словарями и регулярными выражениями, а доразметка и правка осуществлялась в ручную.

На текущий момент нам удалось разметить порядка 1000 описаний вакансий, на основании которых мы начали эксперименты. Мы планируем продолжать заниматься разметкой, т.к. всего 1000 данных - это сравнительно мало для LLM и качественного решения задачи, но разметка данных это долгий и друдоемкий процесс.

5 Эксперименты

5.1 Метрика

Для оценки качества модели в нашем проекте были выбраны основные метрики классификации.

1. Точность (Accuracy): Точность представляет собой долю правильно классифицированных образцов от общего числа образцов. Формула:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

где: TP - количество верно предсказанных положительных классов (True Positives), TN - количество верно предсказанных отрицательных классов (True Negatives), FP - количество неверно предсказанных положительных классов (False Positives), FN - количество неверно предсказанных отрицательных классов (False Negatives).

2. Полнота (Recall или Sensitivity): Полнота измеряет способность модели обнаружить положительные образцы. Формула:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

3. Точность (Precision): Точность показывает, насколько точно модель классифицирует положительные образцы. Формула:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

4. F-мера (F1-score): F-мера является сбалансированным средним между полнотой и точностью и используется для обобщенной оценки модели. Формула:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5.2 Постановка эксперимента

В качестве первой модели мы выбрали distilbert-base [Raj et al., 2023] и провели дополнительную настройку, разморозив несколько последних слоев. Мы использовали ее прогнозы в качестве базового прогноза. Однако, далее мы решили исследовать более новые модели, которые могут обладать лучшей производительностью и точностью.

5.3 Baselines

До появления трансформеров, для решения задачи извлечения именованных сущностей (NER) использовались различные подходы и модели. Некоторые из них

1) Правила и регулярные выражения: Этот подход основывается на определении набора правил и шаблонов, которые позволяют идентифицировать и извлекать именованные сущности из текста. Например, можно определить правило, которое говорит, что слово, начинающееся с заглавной буквы и следующее за ним слово, также является именованной сущностью. Однако, этот подход требует тщательной ручной настройки, поэтому может занять очень много времни при этои он не всегда может обрабатывать сложные или разнообразные паттерны.

2) Модели на основе скрытых марковских моделей (Hidden Markov Models, HMM): HMM - это статистическая модель, которая моделирует последовательности и включает скрытое состояние и наблюдаемые состояния. В случае задачи NER, скрытые состояния представляют собой именованные сущности, а наблюдаемые состояния - слова в тексте. Пример исследования представлен [Morwal et al., 2012]

3) CRF (Conditional Random Fields): CRF - это вероятностная графическая модель, которая моделирует условные вероятности меток для последовательности наблюдений. Она широко используется для задач NER, к примеру [Gridach et al., 2017]. CRF модели учитывают контекст информации и обрабатывают задачу как последовательность меток. Они могут использовать различные признаки (например, слова, POS-теги, грамматические свойства) для принятия решений о метках именованных сущностей.

6 Результаты

В качестве результатов проделанной работы, приведен сравнительный график моделей distil-bert и ruBert, которые показали лучший результат из всех протестированных моделей на нашем кастомном датасете. По результатам сравнения моделей RuBert и Distil-Bert по F1-мере, можно заметить, что RuBert показал лучшее качество. Это может быть связано со следующими факторами.

Во-первых, RuBert имеет больший объем разметки, чем Distil-Bert, что позволяет ему лучше улавливать особенности русского языка и более точно распознавать его особенности.

Во-вторых, RuBert был изначально обучен на русском языке, что позволило ему лучше адаптироваться к особенностям языка и более точно распознавать его особенности.

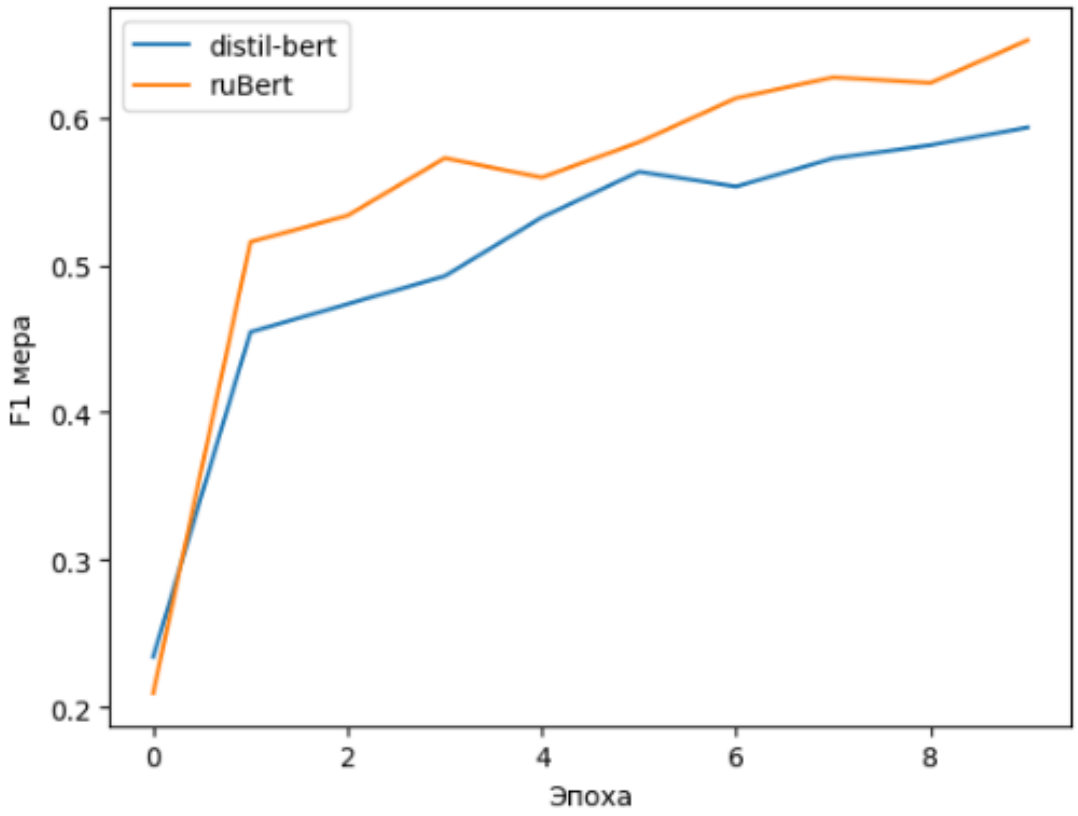

Рис. 7

В таблице 1 представлены значения метрик качества для модели ruBert на валидационной выборке на 10-й эпохе. Из таблицы можно заметить, что наихудшее качество распознавания наблюдается для класса Skills. Возможной причиной такого результата может быть недостаточно точная разметка данного класса.

	precision	recall	f1-score	Support
O class	0.84	0.94	0.89	16732
Salary	0.72	0.92	0.80	968
Skills	0.55	0.52	0.53	3149
Title	0.27	0.71	0.39	1302

Таблица 1: Полученные результаты для модели ruBert

7 Заключение

В ходе нашего исследования задачи NER, мы ознакомились с ее основными этапами и изучили различные подходы, используемые для ее решения.

Однако, в ходе работы мы столкнулись с рядом проблем и вызовов. Изначально мы использовали неправильные подходы к разметки и сбору данных, что в итоге сказалось на качестве обученых моделей. Исходя из этого опыта, мы осознали важность построения качественной разметки.

Тем не менее, мы смогли собрать и разметить достаточное количество данных для обучения моделей на русском языке. Мы провели сравнительный анализ нескольких моделей, что помогло нам оценить их эффективность в решении нашей задачи NER.

Этот опыт позволил нам приобрести практические навыки в работе с данными и моделями для задачи NER. В дальнейшем мы ппланируем улучшить качество разметки данных и попробывать получить модель котрую можно будет лекго использовать на практике для анализа вакансий в телеграмм каналах.

Список литературы

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

[Ding et al., 2021] Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H.-T., and Liu, Z. (2021). Few-nerd: A few-shot named entity recognition dataset.

[Gridach et al., 2017] Gridach, M., Haddad, H., and Mulki, H. (2017). Fner-bgru-crf at cap 2017 ner challenge: Bidirectional gru-crf for french named entity recognition in tweets.

[Morwal et al., 2012] Morwal, S., Jahan, N., and Chopra, D. (2012). Named

entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing*, 1:15–23.

[Nyqvist, 2021] Nyqvist, A. (2021). Bootstrapping annotated job ads using named entity recognition and swedish language models. *Dissertation*.

[Raj et al., 2023] Raj, S., Chitra, P., Silesh, A., and Lingeshwaran, R. (2023). Flood severity assessment using distilbert and ner. pages 391–402.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.