“al qaida” unless we use a low Jaccard similarity threshold of 0.33. Still, “al qaeda” or “al-qa’ida” won’t be matched with “al qaeda” as the Jaccard similarity value is 0. In contrast, edit distance with a threshold 2 can capture all these alternative spellings of the same entity.  

- It may result in too many matches. Set-based similarity measures disregard the order among tokens and dissimilarity between mismatched tokens. For example, “al qaida” will match “al gore” as well as “al pacino” if we use a Jaccard similarity threshold of 0.33. Edit distance with a threshold of 2 won’t report such false positive matches.

To remedy the above problems, we propose to study the entity extraction problem with the well-known string-based dissimilarity measure — edit distance. Edit distance measures the minimum number of edit operations (insertion, deletion, and substitution) to transform one string to another. Edit distance can effectively capture typographic errors, words with alternative spellings, and does not rely on the separation of word boundaries. Hence, edit distance and its variants have been used in named matching [42] and record linkage [7].

A widely adopted method to find approximate matches with an edit distance constraint is based on q-grams [16]. However, a unique feature in named entity matching is the existence of many short entities in the dictionary. Matching short strings approximately results in the following dilemma: we have to use short q-grams to ensure matching strings have at least one common q-gram; however, it is known that short q-grams suffer from poor performance problems [42, 39].

In this paper, we propose to solve the approximate dictionary matching problem with edit distance constraint by an improved neighborhood generation-based method. The neighborhood generation method was traditionally considered only applicable to small alphabet size and small edit errors, as the size of the neighborhood is $O(m^t|E|)$, where $m$ is the string length, $|E|$ is the alphabet, and $t$ is the edit distance threshold [37].Recently, [33] proposed the FastSS algorithm, which reduces the neighborhood size to $O(m^t)$ and was demonstrated to outperform the q-gram-based method on the English vocabulary. However, FastSS still cannot scale up to the diverse entity lengths and error levels for typical entity extraction tasks. In this work, we improve the FastSS method by novel partitioning and prefix pruning techniques and results in a neighborhood size of $O(t_{sp})$, where $t_{sp} \leq t$ is a tunable parameter. Another novelty lies in the document processing algorithm, where we apply a semi-join [6] style reduction technique to avoid considering many unnecessary query and entity pairs, in addition to other optimizations. Experiment results show that the proposed algorithm has superior performance to other alternatives on publicly available named entity recognition datasets with up to 25x speedup.

Note that allowing approximate matching in NER will increase false positive matches. In addition, orthographical matching does not solve the issue of homonyms. In this work, we aim to utilize this approach to increase the recall of the NER systems, following [35, 38].

Additional post-processing methods can be applied to achieve high precision. We also focus on solving the problem exactly, thus excluding approximate (e.g., LSH) or heuristic methods (e.g., BLAST).

Our contributions can be summarized as follows:

- We study the problem of efficiently performing dictionary-based entity extraction with edit distance constraints. It captures an important class of approximately matching entities that is hard to be detected by existing methods based on token-level similarity measures.
- We address the major technical problem in existing neighborhood generation-based algorithms, thus making it a highly competitive method for entity extraction. We devise new partitioning and prefix pruning techniques to reduce the size of the neighborhood from $O(m^t)$ to $O(t_{sp})$.
- We propose an efficient query processing algorithm. The efficiency mainly comes from two facts: we avoid considering unnecessary entity and query segment combinations and we exploit the sharing of computation.
- We have conducted extensive experiments using several named entity recognition datasets in various domains. The proposed method has been shown to outperform other alternatives by up to an order of magnitude.

2. PROBLEM DEFINITION AND PRELIMINARIES

2.1 Problem Definition

DEFINITION 1. Given a document $D$ and a dictionary $E$ of entities, the task of approximate dictionary matching with edit distance threshold $\tau$ is to find all substrings in $D$ such that they are within $\tau$ edit distance from one of the entities in $E$, or more formally, return \[(D[i .. j], E_k) \mid \exists k, \text{ed}(E_k, D[i .. j]) \leq \tau\]

A straight-forward algorithm would be to iterate through all the valid substrings of the document $D[i .. j]$, and issue a similarity selection query to the dictionary to retrieve the set of entities that satisfy the constraint. We refer to each substring as a query segment. As is typical in entity extraction tasks, we do not assume the documents to be matched are given before hand.

Notations. We denote the length of the shortest (longest) entity in the dictionary as $L_{\text{min}}$ ($L_{\text{max}}$). We use $D[i .. j]$ to denote a substring of $D$ that starts at the $i$-th position and ends at the $j$-th position. All array indexes start from 1.

We denote the length of a string $s$ as $|s|$. The $l$-prefix of a string $s$ is its first $l$ characters, i.e., $s[1 .. l]$. We use $|abc...|$ to indicate an ordered sequence of characters.

2.2 Analysis of Previous Approaches

A widely used method for answering similarity selection or join queries with edit distance threshold is to convert the edit distance constraint into a weaker count constraint on matching q-grams. Given a string $s$, we obtain its q-gram multiset by sliding a window of width $q$ over the string. If two strings $s$ and $t$ are within edit distance $\tau$, they must share at least $L_{\text{dist}}$ q-grams, where $L_{\text{dist}} = \max(|s|, |t|) - \tau + 1 - q$. Other filtering criteria, such as length filtering and position filtering, can be incorporated into the above count filtering [16]. An efficient way to find the candidate strings