Paper Review - Auto-FuzzyJoin: Auto-Program Fuzzy Similarity Joins Without Labelled Examples

Peng Li, Xiang Cheng, Xu Chu, Yeye He, Surajit Chaudhuri

Carmel Gafa

April 2, 2025

Fuzzy-join (or similarity join)

Left Table

id	Isem
l1	Peppi Azzopardi
l2	Annetto Depasquale
l3	Karmenu Vassallo

Right Table

id	Isem
r1	Karmnu Vassallo
r2	Ġużeppi Azzopardi
r3	Annetto De Pasquale

Left Table

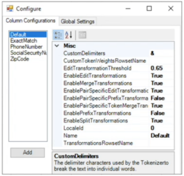
L-id	L-name	L-director	L-description
l1	Carrie	Brian De Palma	Carrie White is shy and outcast ...
l2	Vibes	Ken Kwapis	Psychics hired to find lost temple...

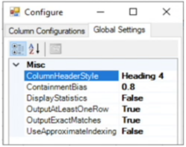
Right Table

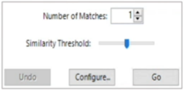
R-id	R-name	R-director	R-description
r1	Carrie	Brian DePalma	This classic horror movie based ...
r2	Vibes	Ken Kwapis	Two hapless psychics unwittingly...

- Fuzzy join takes two tables as inputs and identify record pairs that refer to the same entity
- As an example, l1 and r2 refer to the same person
- The concept can be extended to rows consisting of multiple columns

Fuzzy-join configuration







- Fuzzy-join has been integrated many commercial applications
- These systems are normally not easy to use as they have a big number of configuration parameters
- The extension in Microsoft Excel has 19 options that span across 3 dialog boxes
  - 11 are binary, thus resulting in 2048 possible configuration scenarios
  - 8 continuous, such as thresholds and biases
- In order to execute quality Fuzzy-joins, these configurations must be set up properly by the user

Theoretical foundation: fuzzy join

Given a **reference table** L and a table R containing records that may be **imprecise** or noisy, a **fuzzy join** J establishes approximate matches between them.

- J connects elements of R to similar elements in L based on a chosen **similarity measure** (e.g., Levenshtein distance, cosine similarity, Jaccard similarity).
- Each record $r \in R$ is mapped to at most one record $l \in L$, or **no match at all** (denoted by $\perp$).
- The join is **many-to-one** because multiple records in R can be associated with the **same** record in L, but each $r \in R$ has only **one** possible match.

Formally:

$$J : R \to L \cup \perp$$

Theoretical foundation: fuzzy join configuration space

A **fuzzy join** f compares two strings, r and l, by computing a distance score that reflects their similarity. The computation of this score is governed by a variety of parameters, forming a **parameter space**.

Definition: Each unique combination of these parameters defines a specific **join function** $f \in \mathcal{F}$, where $\mathcal{F}$ is the space of all possible join functions.

Example: Fuzzy Join Distance Score Computation

Join Function: $f = (L, SP, EW, JD)$

- **L**: Lower-casing (Preprocessing)
- **SP**: Space Tokenization
- **EW**: Equal Weights
- **JD**: Jaccard Distance

Inputs:

- $l =$ `"2012 tigers lsu baseball team"`
- $r =$ `"2012 lsu baseball team"`

Tokenization (SP):

- $l \rightarrow \{2012, tigers, lsu, baseball, team\}$
- $r \rightarrow \{2012, lsu, baseball, team\}$

Jaccard Distance:

- $A \cap B = \{2012, lsu, baseball, team\} \rightarrow |A \cap B| = 4$
- $A \cup B = \{2012, tigers, lsu, baseball, team\} \rightarrow |A \cup B| = 5$
- Jaccard Similarity $= \frac{4}{5} = 0.8$
- Jaccard Distance $= 1 - 0.8 = 0.2$

Result: $f(l, r) = 0.2$

Theoretical foundation: threshold and join configuration

- Once the distance $f(l, r)$ is computed:
  - It is compared to a threshold**compared to a threshold** θ to decide whether to join the string pair l and r.
  - If $f(l, r) \leq \theta$, the pair is considered **a match**.
- Together, the function f and the threshold θ define what the authors call a **join configuration**:

$$C = (f, \theta)$$

- This configuration encapsulates both:
  - How distance is computed.
  - When two strings are considered similar enough to be joined.

Definition: A join configuration C is a 2-tuple $C = \langle f, \theta \rangle$, where $f \in \mathcal{F}$ is a join function, and θ is a threshold.
We use $\mathcal{S} = \{\langle f, \theta \rangle \mid f \in \mathcal{F}, \theta \in \mathbb{R}\}$ to denote the space of join configurations.

Theoretical foundation: fuzzy join mapping

Given two tables L and R , a join configuration $C \in \mathcal{S}$ induces a **fuzzy join mapping** J_C , defined as:

$$J_C(r) = \underset{l \in L,\ f(l,r) \leq \theta}{\arg\min}\ f(l, r),\ \forall r \in R$$

That is

- For each record $r \in R$, find $l \in L$ that minimizes the distance $f(l, r)$, **only if** that distance is less than or equal to the threshold θ.
- If no such $l \in L$ exists such that $f(l, r) \leq \theta$, then $J_C(r)$ is **empty** — i.e., no match for that record.

Theoretical foundation: the problem with single join configurations

Real-world data can exhibit **multiple types of variations simultaneously**, such as:

- **Typos**
- **Missing tokens**
- **Extraneous information**

As a result, relying on a **single join configuration** often fails to capture all valid matches, particularly when high **recall** is required.

To handle this diversity, the algorithm uses a **set of join configurations**:

$$U = \{C_1, C_2, \ldots, C_K\}$$

Instead of relying on a single configuration, the system computes join results from each one. This approach allows the system to:

- Accommodate diverse types of variations.
- Improve overall recall by **combining multiple perspectives** on similarity.

L-id	L-Table (Reference Table)		R-id	R-Table (Input Table)
l_1	2008 LSU Tigers baseball team	✓	r_1	2008 LSU baseball team
l_2	2008 LSU Tigers football team	✓	r_2	2008 LSU football team
l_3	2008 Mississippi State Bulldogs baseball team	✓	r_3	2008 Missisippi State Bulldog baseball team
l_4	2008 Mississippi State Bulldogs football team	✓	r_4	2008 Mississippi State Bulldog football team
l_5	...		r_5	...
l_6	2007 LSU Tigers football team	✗	r_6	2007 LSU Tigers baseball team
l_7	2007 Wisconsin Badgers football team	✗	r_7	2008 Wisconsin Badgers football team
l_8	2011 LSU Tigers football team	✗	r_8	2010 LSU Tigers football team
l_9	2007 LSU Tigers baseball team	✗	r_9	2007 LSU Tigers football team

- A **Jaccard distance** with threshold 0.2 works well for pairs like (l_1, r_1), which differ by only one or two tokens.
- However, for pairs like (l_3, r_3) with **spelling variations**, Jaccard similarity is not enough:
  - Jaccard distance $\approx 0.5 \rightarrow$ too high to match under the 0.2 threshold
  - A more suitable metric is **Edit Distance**, which can better align such pairs.

Theoretical foundation: fuzzy join via multiple configurations

- To handle this diversity, the algorithm uses a **set of join configurations**:

$$U = \{C_1, C_2, \ldots, C_K\}$$

- Instead of relying on a single configuration, the system computes join results from each.
- This approach allows the system to:
  - Accommodate diverse types of variations.
  - Improve overall recall by **combining multiple perspectives** on similarity.

Given L and R, a set of join configurations $U = \{C_1, C_2, \ldots, C_K\}$ induces a **fuzzy join mapping** J_U, defined as:

$$J_U(r) = \bigcup_{C_i \in U} J_{C_i}(r), \ \forall r \in R \tag{2}$$

This means that the overall result of the fuzzy join using configuration set U is the **union** of results from all individual configurations $C_i \in U$.

Each configuration $C_i \in U$ is designed to capture a **specific type of string variation** (e.g., typos, missing tokens, extra tokens).
Two records are considered **joined by the set U if and only if** they are joined by **at least one** configuration $C_i \in U$.

- Each configuration contributes **high-quality joins** targeted at particular data challenges.
- The overall join is more **robust and comprehensive**.