Use Case—Nostro Accounts Match

Volker Liermann, Sangmeng Li, and Johannes Waizner

1 Introduction

Cost pressure in the banking industry is, as of today (2021), ever-present. Some issues originated a long time ago, like the low interest rates and their impact on profitability. Other aspects are caused by singular events (such as the COVID-19 pandemic) and their impact on higher impairments in the loan portfolios. New players have entered the marketplace: Fintech companies, but also the Big Tech companies (GAFAM[1]). They cut certain parts of the banking value chain and therefore reduce the margins.

It is expected that these trends will continue in the years to come. Cost pressure is going to rise even more. Thus, there is a strong imperative to seek efficiency improvements and cut costs.

[1] Google, Apple, Facebook, Amazon, and Microsoft (and in Asia Tencent and Alibaba).

V. Liermann (✉) · S. Li · J. Waizner
ifb SE, Grünwald, Germany
e-mail: Volker.Liermann@ifb-group.com

S. Li
e-mail: Sangmeng.Li@ifb-group.com

J. Waizner
e-mail: Johannes.Waizner@ifb-group.com

V. Liermann and C. Stegmann (eds.), *The Digital Journey of Banking and Insurance, Volume II*, https://doi.org/10.1007/978-3-030-78829-2_2

RPA (see Czwalina et al. 2021; Soybir 2021) is a pragmatic way to automate simple and recurring processes. Process mining can reveal the parts of a process worth a deeper analysis.

The process of matching nostro accounts has long been known and—due to its repetitive nature combined with a high frequency (daily)—was a target for optimization long before the digitalization hype began. Matching the two lists (incoming payments and expected payments) is a challenge that has only been solved by IT to a certain extend. Except for trivial cases, the matching itself remained an intellectual challenge for a real person.

1.1 General Idea

Matching by nostro account is tricky, because a brute force approach[2] is—due to computational complexity—only feasible for a small number of payments. A brute force approach would in addition not use the text for identifying the payments.

This chapter shows a matching approach using machine learning. The two-step approach is a combination of a clustering algorithm and a brute-force-style match. The clustering algorithm is used to identify and narrow the possible candidates down to a reasonable number (step one). Within the second step, the remaining (reduced number of) candidates are matched.

1.2 Structure of the Chapter

The second section introduces the business requirements and the origin of the nostro account match. The next section offers tools and algorithms to solve the challenge of the nostro account match by similarity analysis. The section closes with an example application to a dataset. In Sect. 4, we highlight the potential of NLP in matching business events and billing information (billing text and amount). The fifth section summarizes the findings.

[2] Trying out all possible combinations of expected and incoming payments.

2 Business Requirements

2.1 Correspondent Bank System

To better understand the task of matching nostro accounts, this section introduces the purpose and the key elements of the correspondent bank system.

Receiving and paying in a foreign country with a domestic currency is tricky if the bank does not have the required infrastructure and a local bank license. If the bank acts as an affiliate or branch in the foreign country, it works together with a local bank (correspondent bank) to settle with local clients and the foreign currency. Often—due to the affiliate or branch status—the bank does not even have direct access to the foreign central bank settlement system.

Example
A German automotive company (customer of Deutsche Bank) buys steel from a steel manufacturer who is a customer of HSBC for $25 million. Deutsche Bank holds its dollars at The Bank of New York Mellon and HSBC holds its dollars at Bank of America. When the German automotive company instructs its bank to pay the money, HSBC debits the steel manufacturer's account and transfers dollars from its account at BofA to Deutsche Bank's account at BNY. Then, HSBC credits the dollars to the German automotive company's dollar account in Frankfurt.

The accounting terms nostro and vostro (Italian, nostro and vostro; English, "ours" and "yours") are used to differentiate accounts held by a bank for another bank or corporate from an account another bank or corporate holds. A nostro account is held by the other bank. It is an account with our money. Whereas a vostro account holds another bank's money. Therefore, it is an account held by us for another bank or corporate.

In the balance sheet, a debit balance on a nostro account is counted as a cash asset. While a credit balance (i.e., a deposit) on a vostro account is seen as a liability, a debit balance (a loan) on a vostro account is correspondingly taken as an asset.

To keep track of the bank's money being held by the other bank, the nostro account is the right tool. To operationalize, the movements are posted in a shadow account (see Fig. 1). The operational challenge is to keep the shadow account and the nostro account aligned. The alignment is complicated because the booking texts often do not allow a simple one-to-one match of the expected business events and the reported movements on the nostro

account. Often—due to manual processing—small alterations occur. Thus, an exact string match is not a promising approach.

Figure 2 shows the different matching situations from simple (top left 1:1) to complex (bottom right—N:M). N is the number of invoice statements and M is the number of incoming payments.

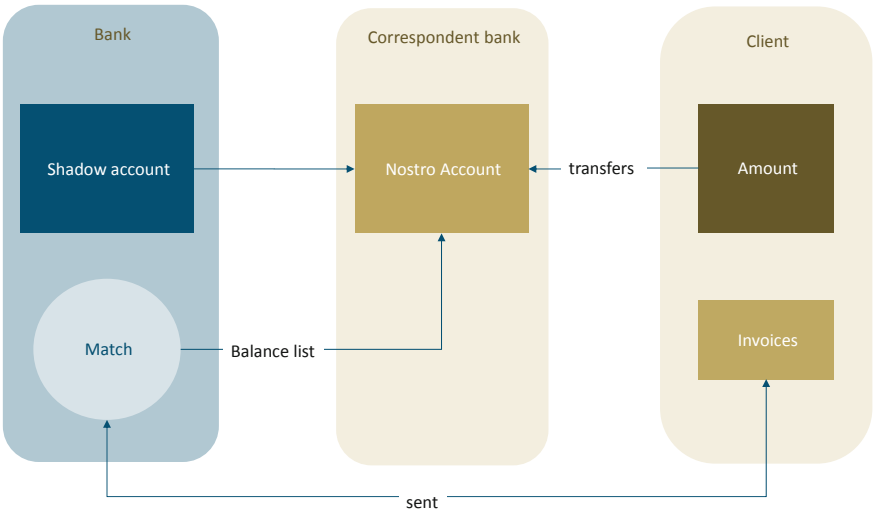

Fig. 1 Involved parties (© ifb SE)

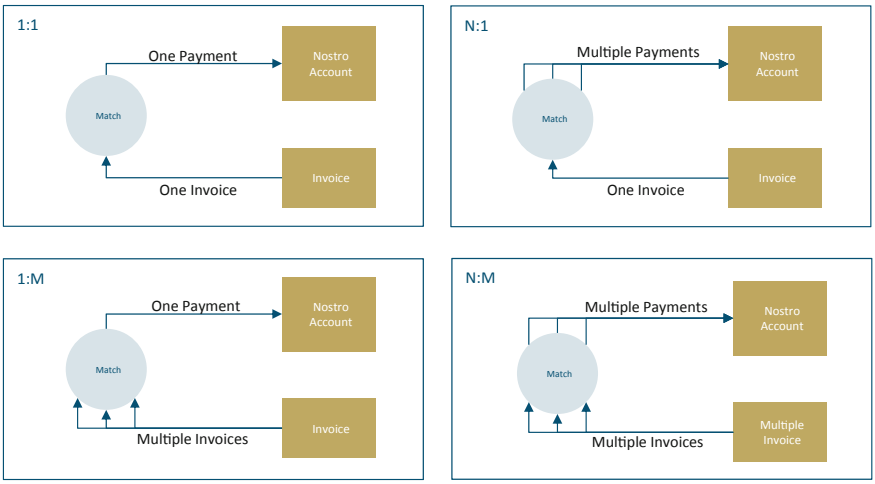

Fig. 2 Different matching situations (© ifb SE)

3 Match Analysis

3.1 Data Collection/Generation

In this chapter we used generated data, which was oriented toward real matching data from project experiences. The data was generated following the different situations in Fig. 2. For each of the situations mentioned (1:1, N:1, 1:M, N:M), we generated several perfectly matching datasets. The task of finding a match was quite easy by using the standard algorithms.

We then contaminated the data by adjusting the text fields, and to make the data like real data we even made small and big changes in invoice and payment numbers. Identification became more challenging, but with some more advanced machine learning models it worked well.

3.2 Brute Force Search

The most common approach in solving match problems such as that mentioned is to use brute force search. This is a very general technique and involves examining all possible combinations and checking whether the amounts match or not. It is simple to implement but usually suffers performance problems, since the computation time grows in linear proportion to the number of data candidates. In this article, we are going to use the power of machine learning algorithms to speed up brute force search. To be more specific, the data candidates are prefiltered according to a match probability, which is predicted by a machine learning model. The number of data candidates can be reduced by deleting the irrelevant candidates or the candidates which are probably not matched.

3.3 Data Analysis/String Similarity

The largest challenge is to figure out how to predict the match probability between each invoice and payment given in Fig. 3. We achieve this by analyzing the text content of column "Text." In this section, we will firstly focus on the direct text analysis, in fact the string similarity. In the next section, we will introduce some advanced extensions in which the content similarity is analyzed based on natural language processing (NLP).

There are a lot of mathematical metrics for measuring string similarity. In the following, we consider the two most common ones: the Levenshtein

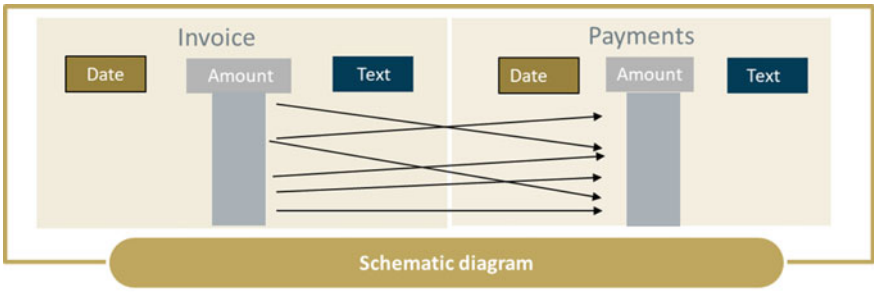

Fig. 3 Brute force search (© ifb SE)

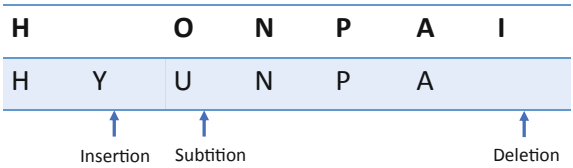

Fig. 4 Levenshtein distance (© ifb SE)

distance (LSD) and the longest common substring distance (LCSD) (Alberto Apostolico 1997).

- Levenshtein distance

The Levenshtein distance measures the difference between two strings by counting the minimum number of single-character edits required to change one into another, where character insertion, deletion, and substitution are considered character edits. As presented in the following example, the Levenshtein distance between HONPAI and HYUNPA is equal to 3, since three-character edits are required (Fig. 4).

- Longest common substring distance

The length of the longest common substring is used as a similarity measure. In comparison to the Levenshtein distance, it is the edit distance only according to insertion and deletion. Again, using the example above, the LCSD between HONPAI and HYUNPA should be equal to 4 (Fig. 5).

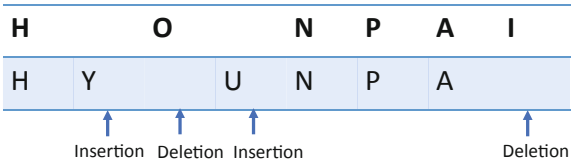

Fig. 5 Longest common substring distance (© ifb SE)

Table 1 Data example of match = 1 (© ifb SE)

	Invoice Text	Payments Text	Match
1	Invoice number: 500537	Ref. 500537	1
2	Invoice number: 500284	Ref: 500284	1
3	Invoice number: 500562	Ref: 500562	1
4	Invoice number: 500800	Ref: 500800	1
5	Invoice number: 500252	Ref: 500252	1
6	Invoice number: 500435	Ref: 500435	1
7	Invoice number: 500438	Ref: 500438	1

Table 2 Data example of match = 0 (© ifb SE)

	Invoice Text	Payments Text	Match
1	Invoice number: 500537	Ref. 500284	0
2	Invoice number: 500537	Ref: 500562	0
3	Invoice number: 500537	Ref: 500800	0
4	Invoice number: 500537	Ref: 500252	0
5	Invoice number: 500537	Ref: 500435	0
6	Invoice number: 500537	Ref: 500438	0
7	Invoice number: 500264	Ref: 500284	0

3.4 Model Training/Features Selection

The following results are implemented in R by using *stringdist* Library (cran.r-project.org 2020) and *h2o* Library (H2O.ai 2019). First, we reconstruct the invoices and payments pairwise as given in the following two tables, where a column "Match" is inserted. This column is binary and shows whether the pair of invoice and payment matches or not (1—match, 0—not match) (Tables 1 and 2).

For each pair of invoice and payment, we compute both Levenshtein distance and longest common substring distance and illustrate them in the following figure, where the blue points stand for matched pairs and the yellow ones for not matched pairs, separately. It is not hard to see that we are not able to achieve a sharp separation by using only one distance. For instance,

the points included in the light blue dashed rectangle are hardly separated into blue and yellow classes if we only use the projection on LCSD. The same for the points in the yellow dashed rectangle if only the Levenshtein distance is used (Fig. 6).

This indicates that the classifier constructed based only on a single distance might suffer underfitting and we are able to build a well-performed classifier by combining both distances. The result can be verified by processing cross-validation for different classifiers, where the result is collected in Table 3. We see that the last classifier has a much better performance than each single distance-based classifier. This best classifier will be used in the next section to predict the match probability. The performance indexes used, AUC, Precision, and Recall, are introduced in another article in this book (Liermann and Li 2021).

In Sect. 3.3, we will introduce a natural language processing technology for computing the text similarity in practical application. The features selection introduced above can be carried out to decide which metrics are relevant and to find the best classifier.

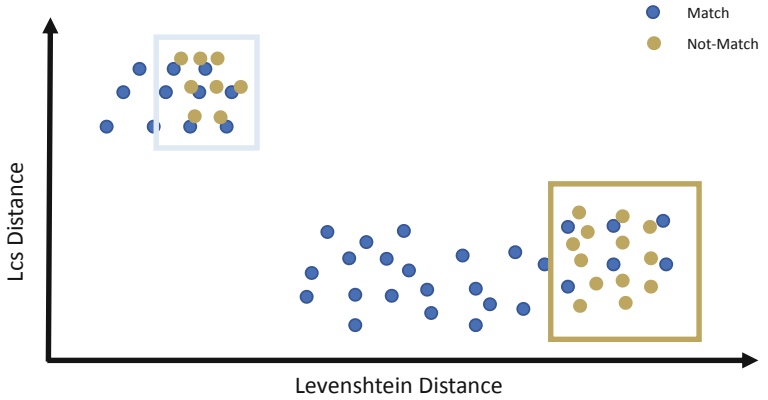

Fig. 6 Levenshtein distance (© ifb SE)

Table 3 Ten cross-validation results (© ifb SE)

Selected features for classifier	AUC (area under the curve)	Precision	Recall
Levenshtein distance	0.9095	0.7345	0.4115
Longest common substring distance	0.8974	0.7545	0.4041
Both distances	0.9972	0.9749	0.9922

3.5 Example of a Match Result

Based on the machine learning model (classifier), we can predict whether a pair of invoice and payment matched or not. In addition, we can also predict the match probability (score). In Table 4, we provide a short example of predictions, where ten pairs of invoice and payment are predicted. The column "Predict" is binary and shows whether the pair matches or not. The remaining two columns indicate the not-match and match probability (p0—not match, p1—match). Not surprisingly, the sum of the last two columns is always equal to 1 by row.

We prefiltered (Fig. 7) the invoices and payments by using the predicted match probability, where the top N candidates with the highest probability can be selected. After that, a brute force search is carried out across the prefiltered candidates. On the one hand, prefiltering/preselection allows us to reduce the number of match candidates and therefore can speed up the brute force search. On the other, it has the risk of losing match accuracy by filtering out matched candidates due to false negative prediction. Therefore, the choice of N plays the key role. In the case of large N, we guarantee a high level of accuracy, but computation time increases. In contrast to that,

Table 4 Prediction (© ifb SE)

Predict	P0	P1
0	0.9414140	0.0585860
0	0.8016686	0.1983314
0	0.8055366	0.1944634
0	0.8016686	0.1983314
1	0.5655805	0.4344195
0	0.8016686	0.1983314
1	0.5655805	0.4344195

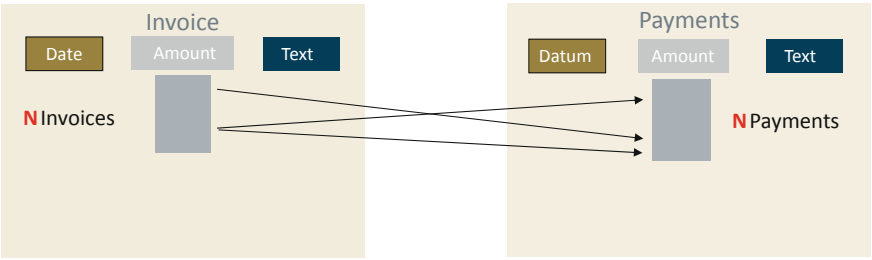

Fig. 7 Prefiltering of invoices and payments: select the top N invoices/payments with the highest match probability (© ifb SE)

Table 5 Example of result with no matched pairs (No.Match = 0) (© ifb SE)

No.Match	Type	Date	Identifier	Amount	Text
0	Invoices	01/01/2020	1:1_1	15,700.6	Invoice number: 500537
2	Invoices	01/01/2020	1:1_2	92,649	Invoice number: 500284
2	Payments	01/01/2020	1:1_2	92,649	Ref: 500284
3	Invoices	01/01/2020	1:1_3	39,328	Invoice number: 500562
3	Payments	01/01/2020	1:1_3	39,328	Ref: 500562
0	Payments	01/01/2020	1:1_1	15,690	Ref: 500537

reducing the value of N will speed up the brute force search but increase the risk of losing accuracy.

After processing the brute force, the match result is exported into a CSV file, where a MatchID is assigned to each matched pair. In addition, the invoices and payments for which no matches were found are automatically moved to a pool with MatchID 0 and need to be processed manually (Table 5).

4 Challenges in the Practical Application—NLP

Natural language processing (NPL) is a field of machine learning which aims to give models the ability to read and understand human languages. It can be used in various business areas, for example, sentiment analysis, which analyzes the customer's choices and decisions based on their social media posts. By using NPL, we can extend our use case into a more general case where the text similarity is no longer based on string similarity but keyword similarity or indeed content similarity. Some examples can be found in (Schröder and Tieben 2021).

5 Summary

Although most banks have automated their nostro account matching to a certain extent, many do not have a fully automated matching process. The reasons for partially manual steps are the complexity of different business events. Alterations and situations occur when packaging one or more business events to one booking or by posting one or more bookings to one business

event. More complexity is added when a combination of the two preceding situations occur.

String similarity algorithms (near match) and tools like the Levenshtein distance (LSD) and the longest common substring distance (LCSD) are useful in solving most of the challenges and improving the level of automation of tasks that still have to be performed manually to some extent.

Literature

Alberto Apostolico, Zvi Galil. 1997. *Pattern Matching Algorithms.* Oxford: Oxford University Press.

cran.r-project.org. 2020. *stringdist: Approximate String Matching, Fuzzy Text Search, and String Distance Functions.* October 9. Accessed December 17, 2020. https://cran.r-project.org/web/packages/stringdist/index.html.

Czwalina, Marie Kristin, Chiara Jakobs, Christopher Schmidt, Matthias Jacoby, and Sebastian Geisel. 2021. "Processes in a Digital Environment." In *The Digital Journey of Banking and Insurance, Volume II—Digitalization and Machine Learning*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.

H2O.ai. 2019. *h2o.ai Overview.* January 29. Accessed October 29, 2020. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html.

Liermann, Volker, and Sangmeng Li. 2021. "Methods of Machine Learning." In *The Digital Journey of Banking and Insurance, Volume III—Data Storage, Processing, and Analysis*, edited by Liermann Volker and Claus Stegmann. New York: Palgrave Macmillan.

Schröder, Daniel, and Marian Tieben. 2021. "Sentiment Analysis for Reputational Risk Management." In *The Digital Journey of Banking and Insurance, Volume II—Digitalization and Machine Learning*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.

Soybir, Sefa. 2021. "Project Management and RPA." In *The Digital Journey of Banking and Insurance, Volume I—Disruption and DNA*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.