Towards Person Name Matching for Inflective Languages

Jakub Piskorski
Joint Research Centre
of the European Commission
Via Fermi 1
21020 Ispra, Italy
Jakub.Piskorski@jrc.it

Karol Wieloch
Poznań University
of Economics
al. Niepodległości 10
60-967 Poznań, Poland
K.Wieloch@kie.ae.poznan.pl

Mariusz Pikuła
Polish-Japanese Institute
of Information Technology
Koszykowa 86
02-008 Warsaw, Poland
s3065@pjwstk.edu.pl

Marcin Sydow
Polish-Japanese Institute
of Information Technology
Koszykowa 86
02-008 Warsaw, Poland
msyd@pjwstk.edu.pl

ABSTRACT

Web person search is one of the most common activities of Internet users. Recently, a vast amount of work on applying various NLP techniques for person name disambiguation in large web document collections has been reported, where the main focus was on English and few other major languages.

This paper reports on knowledge-poor methods for tackling person name matching task in Polish, a highly inflected language with complex person name declension paradigm. These methods apply mainly well-established string distance metrics, some new variants thereof, automatically acquired simple suffix-based lemmatization patterns and some combinations of the aforementioned techniques. Results of numerous experiments are presented.

Categories and Subject Descriptors: H.4.m [Information Systems]: Miscellaneous I.6 [Computing methodologies]: Artificial Intelligence I.7 [Computing methodologies]: Document Processing

General Terms: Algorithms, Languages

Keywords: person name matching, processing highly inflected languages, string distance metrics

1. INTRODUCTION

Finding information about people in the World Wide Web is one of the most popular activities of Internet users. However, the major problem with personal names is that they are not unique and sometimes even for one name many variations exist. Variations may be caused by permutations (e.g., *Simon Perez* and *Perez Simon* might refer to the same person), abbreviations (e.g., *Jan Maria Rokita* may become *J. M. Rokita*), spelling mistakes (e.g., *George Bush* vs. *George Buhs*), usage of accents and foreign characters (e.g., *Schaeffer*, *Schaffer* and *Schäffer*), different transcriptions (e.g. *Jakub*, *Jacob*, *Giacomo* may refer to the same person), postfixes (e.g., names may end with a title like *Jr.* or a number - *John Paul II* vs. *John Paul*), declension paradigm (e.g, *Władimirze Putinie* might be a locative form of *Władimir Putin* in Polish), and other factors. In a multilingual data repository like Web the number of variants for a single person name may quickly rise to couple of hundreds [23].

The task of person name matching is to find synonym and homonym personal names in a given dataset, e.g. Web. Various research communities, ranging from artificial intelligence to databases, have reported on a vast bulk of work on tackling this problem, under a variety of terms such as name disambiguation [20, 16, 5], record linkage [9], duplicate detection [8, 2], or merge/purge [12]. Up to now, the research in this area focused mainly on English texts [1, 7] and few other major languages. Nevertheless, even considering only English web pages most commercial search engines frequently return for a given person name search queries either a blend of links to pages referring to different people, who share the same name (e.g. *Michael Jordan*), or just a tiny fraction of all pages referring to the sought-after person. This is mainly due to the aforementioned types of potential name variations and the fact that significant number of person names in most of the languages is not unique.

In this paper, we explore knowledge-poor methods for supporting and tackling (full) person name matching task in Polish, a lesser studied language with very rich inflection and complex person name declension. In particular, the proposed methods utilize mainly well-established string distance metrics, some new variants of the latter ones, and automatically acquired suffix-based lemmatization patterns. Further, we also investigated whether better accuracy can be obtained via merging different techniques, e.g. combing string distance metric with lemmatization patterns, etc. Results of numerous experiments carried out on a Polish person-name dataset extracted from a Web news corpus are described. We believe that the results presented in this paper could be of importance to solving the same problem for other highly inflective languages, e.g., for most other Slavonic languages (over 400 million speakers).

Our work was mainly inspired by the comprehensive studies on using string distance metrics for name matching tasks presented in [5, 6, 3]. The main motivation of carrying out this research is the fact that processing highly inflective languages adds another complication to the person name matching task. Further, name matching is of paramount importance in Europe Media Monitor system developed by

NLPIX 2008, April 22, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

the Joint Research Centre of the European Commission[1], which aims at aggregating and linking news articles from different sources in 35 languages.

The intuitive way of tackling the inflection problem in Polish and languages which similar inflection paradigm, would be to lemmatize person names, and then to apply string-distance techniques, which turned out to work fine for inflection-poor languages like English. One could argue that the set of inflectional suffixes of names in Polish is finite and the description of combinatorial constraints between such suffixes and corresponding stems is not out of reach. Unfortunately, the person name declension paradigm in Polish is extremely complex and knowledge intensive. Accuracy figures of reported knowledge-based lemmatization systems do not exceed 76%.

As reported by other authors, the inflection in name matching tasks has been dealt with in two ways. The first approach is based on converting names into some kind of canonical form via stripping off inflectional suffixes [4] or truncating all letters after the first k letters of a name [14] (in most languages the inflections are affixed to the end of a word stem with some possible minor alternation of the stem at the junction) and normalizing language specific diacritics (e.g., converting *ä* into *a* in German). In the second step the canonical forms can be used for matching names by using conventional techniques, e.g., string distance metrics [5, 6]. The second approach, reported for instance in [25], is based on generating all possible inflected morphological variants of a given person name in order to capture all potential named mentions of the same person. Although, over-generation inflection forms does not pose a problem (non-existing names would not be matched), such approach requires that base forms are known, which in general might not be the case.

The aim of the work described in this paper was not to provide a fully-fledged solution to person name matching for Polish and related languages, but to explore whether application of knowledge-poor and approximative methods based on string-distance metrics might be useful in the whole process of name matching. In particular, it can be seen in a way as complementary to the previous work mentioned earlier in this paper, i.e., [25], [5], [4] and [14]. However, it is important to note that we did not investigate the context person names appear in, but considered only matching given person names (possibly inflected) against a set of other names, which might be seen as the first step of name matching in textual collections, i.e., collecting documents which might refer to persons with the same name.

The organization of the paper is as follows. First, in 2 we describe the phenomena which complicate person name declension in Polish. In section 3 we briefly report on accuracy figures achieved by some knowledge-based systems for person name lemmatization for Polish, which demonstrates the hardness of the task and shows that there is a lot of space for improvement. Next, in section 4 an overview of string distance metrics and their modifications, which were used in our study, is given. The test data, evaluation methodology and the results of numerous experiments on using string-distance metrics, statistically learned inflection suffixes and combination of the latter two are described in section 5 and 6 resp. The results are discussed in section 7. Finally, we end with a summary and present perspectives for future work in section 8.

[1]http://press.jrc.it/overview.html

Table 1: Declension of Polish male vs. female names

case	male name	female name
nom	*Stanisław Polak*	*Stanisława Polak*
gen	*Stanisława Polaka*	*Stanisławy Polak*
dat	*Stanisławowi Polakowi*	*Stanisławie Polak*
acc	*Stanisława Polaka*	*Stanisławę Polak*
ins	*Stanisławem Polakiem*	*Stanisławą Polak*
loc	*Stanisławie Polaku*	*Stanisławie Polak*
voc	*Stanisławie Polaku*	*Stanisławo Polak*

2. PERSON NAMES IN POLISH

Polish is a West Slavonic language with rich nominal inflection: nouns and adjectives are inflected for case, number and gender[2]. Just like common nouns, Polish person names undergo declension but the inflectional paradigm is more complex. In general, both the first and surname can be inflected, e.g., *Marian Kowalski* (nom.) vs. *Mariana Kowalskiego* (gen./acc.). If the surname is also a regular word form, things get more complicated. Whether it can be inflected in such cases depends on several factors, e.g., on the gender of the first name, a category (part-of-speech) and gender of the (common) word used as a surname. For instance, if the surname is a masculine noun, it is inflected only if the first name is also masculine. The declension of the male name *Stanisław Polak* ('Stanislas Pole') and its variant with the female first name *Stanisława* given in Table 1 illustrates this phenomenon. If the surname is an adjective (e.g., *Niski* 'short' - opposite to 'tall'), it is inflected (according to the adjectival paradigm) and agrees in gender with the first name, i.e., male and female last name forms are different (e.g., *Niski* 'Short' (masc.) vs. *Niska* 'Short' (fem.)).

The declension of foreign surnames may strongly depend on their origin, and in particular on the pronunciation. For example, the name *Wilde* is pronounced differently in English and German, which impacts its declension in Polish. If it is of English origin, a nominal declension is applied, i.e., *Wilde'a* (gen.), whereas if it comes from German, an adjective-like declension is adopted: *Wildego* (gen.). Clearly, inferring the origin of a name from the surface string alone can not be done accurately.

Declension of surnames which are also common nouns can be different from the declension of common nouns[3], e.g., the genitive form of the common noun *gołąb* 'dove' is *gołębia*, whereas the genitive form of the surname *Gołąb* is *Gołąba*.

First names present problems too. Foreign masculine first names, whose pronounced version ends in a consonant or whose written version ends in *-a*, *-o*, *-y* or *-i* do in general get inflected (e.g., *Jacques* (nom.) vs. *Jacques'a* (gen./acc.)), whereas names whose pronounced version ends in a vowel and are stressed on the last syllable (e.g., *François*) usually do not change form. For female first names created from a male first name there is a frequent homonymy between the nominative form of the female name and the genitive/accusative form of the corresponding male form, e.g., *Józefa* is nominative of *Józefa* (fem.) and genitive/accusative of *Józef* (masc.).

[2]There are 7 cases, 2 numbers and 3 genders.

[3]The declension of such surnames depends on the local tradition and sometimes can be identical with the pattern used for common nouns.

To give a final example of the complicacies, consider the person name *Marka Belki*. The first name *Marka* could be either interpreted as a genitive form of the male name *Marek* or *Mark* (foreign version of *Marek*), or as a nominative form of a foreign female name *Marka*. As for the last name *Belki*, it is a genitive form of the common Polish noun *belka* 'beam', but due to the fact that inflection of proper names differs from that of common nouns, we cannot exclude the special proper name form *Belki*. Consequently, there are 6 potential base forms for *Marka Belki*, namely: *Marek Belka* (masc.), *Marka Belka* (fem.), *Marek Belki* (masc.), *Marka Belki* (fem.), *Mark Belki* (masc.), *Mark Belka* (masc.). Even considering the document-level context of the occurrence of the name *Marka Belki* might not be sufficient for resolving the base form ambiguity [21].

A comprehensive overview of this rather intriguing declension paradigm of Polish names is given in [11].

3. PERSON NAME LEMMATIZATION WITH KNOWLEDGE-BASED SYSTEMS

We have carried out some initial experiments on applying existing knowledge-based systems for lemmatization of person names.

In our first experiment, we have tested STEMPELATOR [27] a full-form lexicon-based lemmatizer, which uses a bunch of heuristics for guessing base forms of words not found in the lexicon. To be more precise, we have applied STEMPELATOR on each part of the name (first name, surname) separately. Although STEMPELATOR performs relatively well for common words, the accuracy achieved with the datasets described later in this paper in section 5 were not better than 35% (in case of considering the first result returned by STEMPELATOR). Considering all combinations of the results (base form candidates) returned for the first name and the surname yielded 61% accuracy, which still leaves a lot of space for improvement.

In the second experiment we have tested a more complex and time-intensive system dedicated to person name recognition and person name lemmatization for Polish [21], which exploits: (a) a dictionary of circa 6000 most frequent Polish first names and their morphological variants, (b) a set of sure-fire patterns matching most frequent surname suffixes to their corresponding base forms (e.g, *skiego* → *ski*, and (c) a set of more sophisticated rules relying on higher-level linguistic information, which encode most of the types of phenomena described in section 2. In order to evaluate this system, a set of 30 articles (including 856 person names) on various topics (politics, finance, sports, culture and science) has been randomly chosen from *Rzeczpospolita* [28], a leading Polish newspaper. From the set of recognized person names, only 75.6% have been lemmatized correctly (the correct base form was in the set of candidate base forms returned by the system). It is important to note that for 12.4% of the recognized person names more than one base form was returned. The detailed description of the aforementioned experiment is presented in [22].

The observations learned from the two aforementioned experiments were our main motivation for studying whether utilization of string distance metrics, other knowledge-poor techniques, and amalgamation of such methods with the systems like the first one mentioned in this section would yield comparable or better accuracy of lemmatization and person name variant matching for Polish.

4. STRING DISTANCE METRICS

In our experiments on using string distance metrics for the name matching task and lemmatization we used mainly the metrics applied by the database community for record linkage. The point of departure constitutes the well-known *Levenshtein* edit distance metric given by the minimum number of character-level operations (insertion, deletion, or substitution) needed to transform one string into the other [15]. Further we used an extension of *Levenshtein*, namely *Smith-Waterman* (*SW*) metric [24], which additionally allows for variable cost adjustment to the cost of a gap and variable cost of substitutions (mapping each pair of symbols from alphabet to some cost). We tested two settings for this metric namely, one which normalizes the *Smith-Waterman* score with the length of the shorter string and one which uses for the same purpose the *Dice coefficient*, i.e., the average length of strings compared (*SW-D*). Further variants of the latter metric and other edit distance metrics, e.g., *Needleman-Wunsch*, were not taken into consideration since in our prior experiments [21] they did not perform better than the *Smith-Waterman* metrics. In general, most of the edit-distance metrics can be computed in $O(|s| \cdot |t|)$, where s and t are the two strings being compared.

Good results for name-matching tasks [5] have been reported using variants of the *Jaro* metric [29], which is not based on the edit-distance model. It considers the number and the order of the common characters between two strings. Given two strings $s = a_1 \ldots a_K$ and $t = b_1 \ldots b_L$, we say that a_i in s is *common* with t if there is a $b_j = a_i$ in t such that $i - R \leq j \leq i + R$, where $R = \lfloor \max(|s|, |t|)/2 \rfloor - 1$. Further, let $s' = a'_1 \ldots a'_K$ be the characters in s which are common with t (with preserved order of appearance in s) and let $t' = b'_1 \ldots b'_L$ be defined analogously. A *transposition* for s' and t' is defined as the position i such that $a'_i \neq b'_i$. Let us denote the number of transposition for s' and t' as $T_{s',t'}$. The *Jaro* similarity is then calculated as:

$$J(s,t) = \frac{1}{3} \cdot \left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - \lfloor T_{s',t'}/2 \rfloor}{|s'|}\right)$$

A *Winkler* variant of *Jaro* metric boosts this similarity for strings with agreeing initial characters and is calculated as:

$$JW(s,t) = J(s,t) + \delta \cdot boost_p(s,t) \cdot (1 - J(s,t))$$

,where δ denotes the common prefix adjustment factor (default value is 0.1) and $boost_p(s,t) = \min(|lcp(s,t)|, p)$. Here $lcp(s,t)$ denotes the longest common prefix between s and t. For multi-token strings we extended $boost_p$ to $boost^*_p$. Let $s = s_1 \ldots s_K$ and $t = t_1 \ldots t_L$, where s_i (t_i) represent i-th token of s and t resp., and let without loss of generality $L \leq K$. $boost^*_p$ is calculated as:

$$boost^*_p(s,t) = \frac{1}{L} \cdot \sum_{i=1}^{L-1} boost_p(s_i, t_i) + \frac{boost_p(s_L, t_L..t_K)}{L}$$

We denote the metric which uses $boost^*_p$ as JWM. The time complexity of '*Jaro*' metrics is $O(|s| \cdot |t|)$.

The *q-gram* metric [26] is based on the intuition that two strings are similar if they share a large number of character-level q-grams. We used a variant thereof, namely so called *skip-gram* metric [13]. It is based on the idea that in addition to forming bigrams of adjacent characters, bigrams that skip characters are considered. *Gram classes* are defined that specify what kind of skip-grams are created, e.g. $\{0,1\}$ class means that normal bigrams are formed, and bigrams that skip one character. This metric can be computed in $O(\max\{|s|,|t|\})$. Our previous experiments showed that it outperforms the classic *q-gram* metric and suchlike metrics, e.g., (*positional q-grams*), which takes into account only common q-grams that occur within a maximum distance to each other [10].

Considering the declension paradigm of Polish we also considered a basic and time efficient metric based on the longest common prefix information, which would intuitively perform well in the case of single-token names.[4] It is calculated as: $CP_\delta(s,t) = (|lcp(s,t)| + \delta)^2/|s| \cdot |t|$. The symbol δ in $CP_\delta(s,t)$ is an additional parameter for favouring certain suffix pairs in s (t). We have experimented with two variants, CP_{δ_1} and CP_{δ_2}. In CP_{δ_1} the value of δ is set to 0. In CP_{δ_2}, as a result of empirical study of the data and the declension paradigm δ has been set to 1 if s ends in: *o*,*y*,*ą*,*ę*, and t ends in an a. Otherwise δ is set to 0. For coping with multi-token strings we introduced a new similar metric called *weighted longest common substrings* distance ($WLCS$) - a variant of the better-known *longest common substrings* distance metric, which recursively finds and removes the longest common substring in the two strings compared. Let $lcs(s,t)$ denote the 'first' longest common substring for s and t and let s_{-p} denote a string obtained via removing from s the first occurrence of p in s. The LCS metric is calculated as:

$$LCS(s,t) = \begin{cases} 0 \text{ if } |lcs(s,t)| \leq \phi \\ |lcs(s,t)| + LCS(s_{-lcs(s,t)}, t_{-lcs(s,t)}) \end{cases}$$

The value of ϕ is usually set to 2 or 3. The time complexity of LCS is $O(|s| \cdot |t|)$. In the extended version, i.e., $WLCS$, an additional weighting to the $|lcs(s,t)|$ is introduced. The main idea is to penalize longest common substrings which do not match the beginning of a token in at least one of the compared strings. Let α be the maximum number of non-whitespace characters, which precede the first occurrence of $lcs(s,t)$ in s or t. Then, $lcs(s,t)$ is assigned the weight $(|lcs(s,t)|+\alpha-\max(\alpha,p))/(|lcs(s,t)|+\alpha)$, where p has been experimentally set to 4.

Finally, we tested the recursive schema, known also as *Monge-Elkan* (ME) distance [19]. Let us assume that the strings s and t are broken into substrings (tokens), i.e., $s = s_1 \ldots s_K$ and $t = t_1 \ldots t_L$. The intuition behind *Monge-Elkan* measure is the assumption that s_i in s corresponds to a t_j with which it has highest similarity. The similarity between s and t equals the mean of these maximum scores. Formally, the *Monge-Elkan* metric is defined as follows, where *sim* denotes some secondary similarity function.

$$ME(s,t) = \frac{1}{K} \cdot \sum_{i=1}^{K} \max_{j=1\ldots L} sim(s_i, t_j)$$

Inspired by the multi-token variants of the JW metric presented in [3] we introduced two additional metrics, which are similar in spirit to the *Monge-Elkan* metric. The first one, *Sorted-Tokens* (ST) is computed in two steps. Firstly, the tokens constituting the full strings are sorted alphabetically. Next, an arbitrary metric is applied to compute the similarity of the 'sorted' strings. The second metric, *Permuted-Tokens* (PT) compares all possible permutations of tokens constituting the full strings and returns the maximum calculated similarity value.

5. TEST DATA AND EVALUATION

This section describes the test data and evaluation methodology used in our experiments on using different techniques for the name matching (and lemmatization) task.

We define the problem as follows. Let A, B and C be three sets of strings over some alphabet Σ, with $B \subseteq C$. Further, let $f : A \to B$ be a function representing a mapping of inflected forms into their corresponding base forms. Given, A and C (the latter representing the search space), the task is to construct an approximation of f, namely $\widehat{f} : A \to C$. If $\widehat{f}(a) = f(a)$ for $a \in A$, we say that $\widehat{f}$ returns a *correct answer* for a, otherwise, $\widehat{f}$ is said to return an *incorrect answer*. We say that $\widehat{f}$ returns a *quasi-correct answer* for a if $\widehat{f}(a) = f(a)$ or $f(\widehat{f}(a)) = f(a)$ (the answer is the base form or another variant thereof).

Secondly, we defined an additional task consisting of constructing another approximation of f, namely function $f^* : A \to 2^C$, where f^* is said to return a *quasi-correct answer* for $a \in A$ if $\forall a' \in f^*(a) : f(a) = a' \vee f(a) = f(a')$, i.e., $f^*(a)$ contains only strings which are either the base form of a or a variant of a, e.g., morphological variant.

5.1 Test Data

For the experiments we have used two datasets: (a) a mapping of full person names (first name + surname) to their base forms (PFN-1) consisting of 1548 pairs[5], and (b) another variant of the latter one with some hard-to-tackle cases (e.g., inverted order of first name and surname) and consisting of 1538 entries (PFN-2). The aforementioned resource were created semi-automatically as follows. We have automatically extracted a list of circa 22952 full person-name candidates from a corpus of 15,724 on-line news articles from the *Rzeczpospolita* corpus [28], via using first name lexicon consisting of over 6000 most popular Polish first names (including their morphological variants) and an additional list of 58038 uninflected foreign first names. Subsequently, we have selected an excerpt of circa 1900 entries (inflected forms) from this list. 1/3 of this excerpt are the most frequent names appearing in the corpus, 1/3 are the most rare names, and finally 1/3 of the entries were chosen randomly. Finally this list was cleaned and duplicates were removed. The full set of the person name candidates was extended in order to include all base forms (22064 entries) and was used as the search space in all experiments.

5.2 Accuracy Metrics

We measured the accuracy in four ways. Firstly, we calculated the accuracy with the assumption that a multi-result answer is incorrect and we defined *all-answer accuracy* (AA) measure which penalizes the accuracy for multi-result answers. Second measure, *all-answer relaxed accuracy* (AAR)

[4]This metric was used as an inner metric in recursive metrics described later in this section since it is not capable 'alone' to accurately match multi-token strings

[5]Pairs, where inflected form is identical with the base form have been excluded from the experiments since in such a case finding an answer is straightforward.

```
CommonPrefix-MostSimilar(s = s1s2, Space)
1  Cand ← ∅
2  for s' = s'1s'2 ∈ Space
3  do if TotalCommonPrefix(s, s') > |s1| + α · |s2|
4        then Cand ← Cand ∪ {s'}
5  return Cand
```

Figure 1: Algorithm CommonPrefix-MostSimilar

α	AA	SR	AAR	RA
0.4	0.689	**0.894**	0.719	0.846
0,45	**0.698**	0.883	0.728	**0.855**
0.5	0.696	0.829	0.738	0.849
0.55	0.696	0.821	**0.740**	0.848

Table 2: Top results for CommonPrefix-MostSimilar

is a relaxed variant of the latter one, where quasi-correct answers are counted as true positives (the answer is either the base form or another variant of the name, e.g., inflectional variant of the base form). Next, we measured the accuracy of single-result answers (*single-result accuracy* - SR) disregarding the multiple-result answers. Finally, we defined somewhat weaker measure *relaxed accuracy* (RA), which is an extension of AAR, and additionally treats a multi-result answer as true positive if all of the returned results are quasi correct (see definition of f^* in the beginning of section 5), i.e., the result set contains solely strings which are base forms or other variants of the given name.

Let s denote the number of strings, for which a single result was returned. Analogously, m is the number of strings for which more than one result was returned. Next, let s_c (s_{qc}) denote the number of correct (quasi-correct) single-result answers returned. Further, let m_{qc} denote the number of quasi-correct multi-result answers. The accuracy metrics are computed as: $AA = s_c/(s+m)$, $AAR = s_{qc}/(s+m)$, $SR = s_c/s$ and $RA = (s_{qc} + m_{qc})/(s+m)$.

The SR and AA accuracy measures were basically defined for evaluating the usefulness of the explored string distance metrics for performing lemmatization, whereas the intuition behind AAR and RA accuracy metrics was to measure the usability for the more general name matching task.

6. EXPERIMENTS

6.1 Baseline Experiment

In our baseline experiment we evaluated a simple method, which for a given name $s = s_1s_2$, where s_1 and s_2 are tokens representing the first name and the surname resp. returns as an answer all names s' in the search space, for which the total length of common prefixes with s is above a certain threshold. The idea is depicted in the pseudo code in Figure 1. The function TotalCommonPrefix(s,s') in line 3 returns the sum of the lengths of common prefixes of s and s'. The parameter $\alpha \in [0, 1]$ is aimed to determine the minimum overlap factor of surnames. The top results obtained with the baseline method on PFN-1 dataset with various α values is given in Table 2.

6.2 Simple String Distance Metrics

In our next experiment we tested the basic non-recursive metrics described in section 4. The results are given in Table 3. *Smith-Waterman* turned out to achieve the best scores in the AA accuracy for both datasets (79.1% and 57.1% resp.), whereas $WLCS$ was the best metric w.r.t. SR accuracy for PFN-1 (84.1%), followed by *Smith-Waterman* metrics. In case of PFN-2 *Smith-Waterman* family of metrics achieved the best results in SR accuracy, although the figures around 60% are not impressive. *Smith-Waterman*

Table 3: The results for simple metrics

Metrics	AA	SR	AAR	RA
PFN-1				
Levenshtein	0,551	0,722	0,664	0,811
Smith-Waterman	**0,791**	0,829	0,879	0,905
Smith-Waterman-D	0,782	0,813	**0,897**	0,931
JW	0,643	0,700	0,761	0,788
JWM	0,758	0,783	0,889	0,917
skip-grams	0,605	0,704	0,749	0,846
LCS	0,586	0,751	0,694	0,851
WLCS	0,692	**0,841**	0,806	**0,968**
PFN-2				
Metrics	**AA**	**SR**	**AAR**	**RA**
Levenshtein	0,386	0,558	0,481	0,593
Smith-Waterman	**0,571**	**0,620**	0,681	0,710
Smith-Waterman-D	0,557	0,592	0,776	0,813
JW	0,475	0,495	0,598	0,620
JWM	0,542	0,560	0,659	0,683
skip-grams	0,430	0,476	0,832	0,906
LCS	0,410	0,487	0,787	0,906
WLCS	0,473	0,548	**0,847**	**0,970**

metrics and JWM achieve the best results in AAR accuracy for PFN-1 (ca. 87.9-89.7%), whereas $WLCS$ performs best for PFN-2 (84.7%) since it can cope best with the inverted order of first name and surname in PFN-2 dataset. Finally, $WLCS$ significantly outperforms all other metrics in RA category for both datasets (96.8% and 97.0% resp.).

The top results obtained with simple metrics on PFN-1 dataset significantly outperform the corresponding scores obtained with the baseline algorithm presented in 6.1, except the SR accuracy, which is higher in case of the baseline algorithm due to the low number of single-answer results.

6.3 Fine-tuning Smih-Waterman Metrics

The results achieved with *Smith-Waterman* metrics, as reported in the previous subsection, are among the best for the both PFN-1 and PFN-2 datasets. Encouraged by these observations we carried out additional experiments in order to optimize their accuracy performance.

Smith-Waterman metric depends on numerous parameters including *MinCost*, *MaxCost* and *GapCost* (default values are: −2.0, 1.0, 0.5, resp.). We applied random search through this 3-dimensional parameter space, repeating the experiment 500 times. Checking only the tiny fraction of the possible parameter settings, resulted in an accuracy improvement for PFN-1 when compared to the default setting. The top accuracy results achieved with $MinCost = -0.55391$, $MaxCost = 0.29161$ and $GapCost = 0.11144$ are presented in Table 4. In the remaining part of this paper we will refer to the 'optimized' versions of these *Smith-waterman* metrics as SW_2 and $SW\text{-}D_2$ resp.

As for the substitution cost matrix, we also experimented with various search heuristics including random search, grid-search, hill-climbing and simulated annealing for searching the parameter space of around 1000 dimensions. Random search method allowed to improve AAR measure by 1.7%

PFN-1				
Metrics	**AA**	**SR**	**AAR**	**RA**
SW	**0.801**	0.833	0.886	0.914
SW-D	0.789	0.819	**0.901**	0.933
PFN-2				
SW	**0.575**	0.611	0.685	0.712
SW-D	0.563	0.591	0.767	0.802

Table 4: The top results for optimized Smith-Waterman metrics. The overall improvements are written in bold.

with respect to the values achieved for the default setting. To be more precise, the top score achieved for the random search through the substitution matrix space of the *Smith-Waterman with Dice Coefficient* metric was: 77.3% (AA), 78.6% (SR), 91.4% (AAR) and 92.6% (RA). Regular grid-search around the best setting did not improve the results significantly. Further, application of simulated annealing for the default setting yielded some insignificant improvement over the default setting. Therefore, we omit the details of the aforementioned experiments.

6.4 Recursive String Distance Metrics

The recursive metrics performed in some settings significantly better. In particular, the *Monge-Elkan* scheme performed best with CP_{δ_2} as internal metric and somewhat worse results were obtained with JWM and CP_{δ_2} as internal metrics. The 10 top results in all accuracy categories are summarized in Table 5. As for PFN-1 dataset, an improvement of circa $4-5\%$ could ba achieved for AA, AAR and SR when compared to the top results for the basic metrics. In case of PFN-2, the somewhat more 'hard' dataset, the top result in AA and SR accuracy are only slightly better $(1,3\%$ and $0,1\%$ resp.). However, top AAR accuracy is by circa 10% higher.

Table 5: The results for recursive metrics

PFN-1				
Metric	**AA**	**SR**	**AAR**	**RA**
ME & CP_{δ_2}	**0,846**	**0,883**	**0,933**	**0,967**
ME & CP_{δ_1}	0,802	0,850	0,915	0,962
ME & JWM	0,785	0,837	0,884	0,937
PT & JWM	0,781	0,828	0,895	0,943
ST & $SW\text{-}D_2$	0,765	0,811	0,879	0,924
ST & JWM	0,760	0,800	0,881	0,923
ST & $SW\text{-}D$	0,756	0,803	0,873	0,917
ME & $SW\text{-}D$	0,749	0,799	0,855	0,903
PT & $SW\text{-}D$	0,746	0,787	0,869	0,911
ME & $SW\text{-}D_2$	0.743	0.789	0.849	0.897
PFN-2				
Metric	**AA**	**SR**	**AAR**	**RA**
ME & CP_{δ_2}	**0,588**	**0,621**	0,929	0,960
ME & CP_{δ_1}	0,556	0,580	**0,941**	**0,962**
ME & JWM	0,549	0,574	0,927	0,951
PT & JWM	0,549	0,574	0,932	0,956
ST & JWM	0,533	0,554	0,922	0,944
ST & $SW\text{-}D_2$	0.533	0.557	0.914	0.935
ST & $SW\text{-}D$	0,525	0,549	0,909	0,930
ME & $SW\text{-}D$	0,524	0,549	0,904	0,926
PT & $SW\text{-}D_2$	0.523	0.544	0.914	0.934
ME & $SW\text{-}D_2$	0.520	0.543	0.901	0.923

6.5 Combining Metrics

The first and obvious way of merging distance metrics is to combine the 'best' metrics in SR accuracy with the 'best' metrics in the AA category. Let us assume, that two metrics m_1 (good in SR) and m_2 (good in AA accuracy) are too be merged. The idea is to first use m_1 and if it returns a single answer, return it, otherwise return the result of application of m_2. The pseudo code of the corresponding algorithm COMBINEDMOSTSIMILAR is given in Figure 2, where s denotes the input string and $Space$ denotes the search space. The function MOSTSIMILAR$(m_1, s, Space)$ returns for the metric m_1 and the string s the most similar string(s)[6] in the search space $Space$.

```
CombinedMostSimilar(m1, m2, s, Space)
  Cand ← MostSimilar(m1, s, Space)
  if |Cand| = 1
    then return First(Cand)
  return MostSimilar(m2, s, Cand)
```

Figure 2: The algorithm CombinedMostSimilar

Application of the algorithm COMBINEDMOSTSIMILAR to PFN-1 revealed that best results in AA accuracy (around $87.0-87.4\%$) could be achieved (unsurprisingly) with *Monge-Elkan* & CP_{δ_2} as m_1 and simple metrics as m_2. In particular, the best result was achieved with JW (87.4%) and JWM (87.34%). Compared to the the recursive metrics an improvement of 2.8% could be observed. Clearly, the top scores for SR were similar as those for recursive metrics, i.e., around 88%. The top result was achieved with *Monge-Elkan* & CP_{δ_2} (m_1) and *Smith-Waterman* (m_2) $(88,3\%)$. The AAR accuracy could be improved by ca. 3.4%. The best scores (96.7%) in this category were obtained with $WLCS$ (m_1) and *Monge-Elkan* & CP_{δ_2} (m_2). Finally, in the RA category, the best results were achieved via combining $WLCS$ (m_1) and *Monge-Elkan* & CP_{δ_2} as m_2 (97.93%).

Similarly, the AA and AAR scores for PFN-2 could be improved (by 2.45% and 2.71% resp.). Again, for AA the best results $(61,25\%)$ were achieved with *Monge-Elkan* & CP_{δ_2} (m_1) and JW or JWM (m_2). As for AAR, many combinations of m_1 being either *Monge-Elkan* & CP_{δ_2} or *Sorted-Tokens* & $WLCS$ or *Permuted-Tokens* & $WLCS$ and m_2 being either JWM or JW or $WLCS$ or *Smith-Waterman* yields a AAR score between 96.1% and 96.81%. In particular, the top score (96.81%) was achieved with *Monge-Elkan* & CP_{δ_2} (m_1) and LCS (m_2). The best SR accuracy for PFN-2 (62.3%) was achieved with *Monge-Elkan* & CP_{δ_2} (m_1) and *Levenshtein* (m_2). Finally, the best RA score was obtained with *Sorted-Tokens* & $WLCS$ (m_1) combined with $SW\text{-}D_2$ as m_2 (98.8%).

Another variant of the algorithm COMBINEDMOSTSIMILAR computes first all strings, whose distance from s is among the first k distance values in the search space (in an ascending order). These strings constitute then the search space for the metric m_2 in the second step. The corresponding pseudo code (COMBINEDMOSTSIMILAR-2) is presented in Figure 3. The method GETKTHDISTANCEVALUE $(m_1, s, Space, k)$ returns the k-th 'least' distance value for the string s in the search space $Space$.

Surprisingly, the application of this variant on PFN-1 did not result in significantly different accuracy figures from those obtained with COMBINEDMOSTSIMILAR. Interestingly,

[6]There is potentially more than one string in the search space, whose distance from s is the smallest.

```
CombinedMostSimilar-2(m1, m2, s, Space, k)
λ ← GetKthDistanceValue(m1, s, Space, k)
Cand ← {s'|dist_m1(s, s') ≤ λ}
return MostSimilar(m2, s, Cand)
```

Figure 3: Algorithm CombinedMostSimilar-2

```
PatternBased(m, s, Space, k)
1  λ ← GetKthDistanceValue(m, s, Space, k)
2  Cand ← {s'|dist_m(s, s') ≤ λ}
3  if |Cand| = 1
4     then return First(Cand)
5  return SelectUsingPatterns(s, Cand)
```

Figure 4: The algorithm PatternBased

top ranking settings in each category involved *Jaro-Winkler*, *Smith-Waterman* and *WLCS* as m_1, and *Monge-Elkan* & CP_{δ_2} as m_2 metric. In particular, the best score in each category was achieved with *WLCS* (m_1) and *Monge-Elkan* & CP_{δ_2} (m_2). See Table 6 for details. Contrary to PFN-1, significant improvement could be obtained with the algorithm CombinedMostSimilar-2 on PFN-2. In particular, the top scores for AA, SR and AAR were improved against the recursive metrics by 6.5%, 9.1%, and 1.2% resp. The top metric combinations are given in Table 7.

Table 6: Top results for CombinedMostSimilar-2, PFN-1

category	AA	SR	AAR	RA
k	2	3	2	2
score	**0.872**	**0.886**	**0.960**	**0.973**

Table 7: Top AA, SR, AAR, and RA results for CombinedMostSimilar-2 on PFN-2 with $k = 3$

PFN-2			
$metric_1$	$metric_2$	**AA**	**SR**
ME & *JWM*	SW_2	**0.653**	**0.712**
PT & *JWM*	SW_2	0.643	0.704
ST & *JWM*	SW_2	0.643	0.702
ST & SW_2	SW_2	0.640	0.695
PT & *WLCS*	SW_2	0,638	0,699
ST & *WLCS*	SW_2	0,638	0,694
PT & *SW-D2*	SW_2	0,635	0,692
ME & CP_{δ_2}	SW_2	0.633	0.689
PFN-2			
$metric_1$	$metric_2$	**AAR**	**RA**
WLCS	*ME* & CP_{δ_1}	**0,953**	0,975
WLCS	*ME* & CP_{δ_2}	0,952	**0,976**
WLCS	*PT* & *JWM*	0,943	0,969
PT & *JWM*	*ME* & CP_{δ_1}	0,943	0,964
ME & *JWM*	*ME* & CP_{δ_1}	0,942	0,964
ST & *JWM*	*ME* & CP_{δ_1}	0,942	0,963

Finally, we experimented with 'merging' the results of various distance metrics via computing a global rank, which is a linear combination of the corresponding distance values. Since the top score achieved in this way with *Monge-Elkan* & CP_{δ_2}, *Monge-Elkan* & CP_{δ_1}, *WLCS*, *SW-D*, and *JWM* did not result in an improvement of the accuracy (AA = 82.9%, SR = 85.0%, AAR = 94.4%, and RA = 96,5%) we droped this line of explorations.

6.6 Pattern-based method

In our next experiment we have explored whether utilization of a simplistic lemmatization model based on automatically acquired suffix-based patterns can improve the accuracy. We have automatically acquired from a large set of training data a set of triples (*TrainedTriples*) of the form (s_{infl}, s_{base}, f), where s_{infl} is a suffix of an inflected word form, s_{base} is a corresponding suffix in the base form for s_{infl}, and f is the frequency of the pair (s_{infl}, s_{base}) in the training data. We considered all pairs of suffixes of length up to 5 characters. The training data consisted of 1093149 noun entries extracted from the morphologically tagged dictionary taken from *Morfologik* project [18]. These suffix-based patterns were then used to select the base form in case of multi-result answers by the given string distance metric. The formal description of the algorithm (PatternBased) is given in Figure 4. In line 5, a call to SelectUsingPatterns method returns for the string s the preferred base form from the list of candidates in the search space $Cand$, via ranking of suffix-based lemmatization patterns, which match s. The pseudo code of the aforementioned method is given in Figure 5. Initially (line 3-4) lemmatization patterns for the first name and surname resp. are created. Subsequently, for each candidate c (line 6), we select from the lemmatization pattern sets the ones which are compatible with c, i.e., the corresponding 'stem' part of the pattern matches with c, and which have the highest rank[7] (call to BestPattern in lines 9-10). Subsequently candidate c is assigned a rank (line 11), which is a linear combination of the rank for the best first-name pattern and the rank of the best surname pattern (in our experiments α and β are set to 0.5). Finally, the candidate with the best rank (or more if there are more with the same rank) is returned (line 13).

The top results for PFN-1 and PFN-2 tested with simple and recursive metrics are given in Table 8. These results were obtained with $k = 1$, i.e., considering only the smallest distance value. Unfortunately, increasing the value of k did not improve the accuracy figures. As can be observed, the suffix-based algorithm turned to perform significantly better for both PFN-1 and PFN-2 in all categories when compared to the the best results obtained for simple and recursive metrics. However, in case of PFN-1 the results are not significantly different from those obtained with CombinedMostSimilar-2, i.e., AA, AAR and RA are slightly better, whereas SR is the same. For PFN-2 the top AA and SR scores obtained with the suffix-based algorithm are worse by 3.9% and 8.3% resp., whereas the AAR% score is better by 3.2% compared to CombinedMostSimilar-2

Next, we have explored whether deployment of CombinedMostSimilar algorithms as the m metric in PatternBased yields any improvement. We call this variant PatternBased-2. The top scores for both datasets are given in Table 9. Only slight improvement could be observed.

Subsequently, we have experimented with the suffix-based patterns in another way, i.e., via replacing the string-distance metric used in the PatternBased algorithm with a can-

[7]The rank is calculated as the frequency of the pattern raised to the power equal to the length of the suffix in the inflected form

```
SelectPatterns(s = s1 s2 ... sn)
  Ptrns ← ∅
  for i ← 1 to n
  do Ptrns ← Ptrns ∪ AllPatterns(si ... sn)
  return Ptrns

AllPatterns(s = s1 s2 ... sn)
  Ptrns ← ∅
  for (s_infl, s_base, f) ∈ TrainedTriples
  do if s_infl == s1 s2 ... sn
        then Ptrns ← Ptrns ∪ {(s_base, f)}
  return SortDescendingByFreq(Ptrns)

SelectUsingPatterns(s, Space)
  first ← getFirstName(s)
  last ← getLastName(s)
  f−Ptrns ← SelectPatterns(first)
  l−Ptrns ← SelectPatterns(last)
  Cand ← ∅
  for c ∈ Space
  do c_first ← getFirstName(c)
     c_last ← getLastName(c)
     p_first ← BestPattern(c_first, f−Ptrns)
     p_last ← BestPattern(c_last, l−Ptrns)
     rank ← α · rank(p_first) + β · rank(p_last)
     Cand ← Cand ∪ {(c, rank)}
  return TopCandidate(Cand)
```

Figure 5: Algorithm for selecting base forms

didate preselection heuristic, which for a given name $s = s_1 \ldots s_k$ (where s_i's denote tokens not characters) accepts only such names $s' = s'_1 \ldots s'_k$ in the search space, for which $|lcp(s_i, s'_i)| \geq |s_i|/2$ for all $i \in \{1, \ldots, k\}$ holds, i.e., the length of the common prefix of each corresponding token in s and s' is at least 50% of the length of the token in s. The tokens constituting the names are sorted alphabetically before the aforesaid heuristic is applied. In this manner, the 'candidate' sets were significantly larger than in the case of applying other string distance metrics. We refer to this algorithm as PatternBased-WithPreselection. All accuracy results for PFN-1 were significantly worse than the best overall scores obtained so far. Clearly, one could not expect to gain anything w.r.t. AAR and RA due to larger candidate sets. Surprisingly, the AA and SR accuracy for PFN-2 could be improved. All figures are given in Table 10.

Table 8: Top results for PatternBased alg.

Metric	AA	SR	AAR	RA
PFN-1				
ME & CP_{δ_2}	**0,882**	**0,886**	0,970	0,971
ME & CP_{δ_1}	0,848	0,851	0,963	0,965
$WLCS$	0,841	0,846	0,972	0,976
PT & $WLCS$	0,838	0,842	0,970	0,974
ST & $WLCS$	0,837	0,842	**0,976**	**0,979**
PFN-2				
ME & CP_{δ_2}	**0,614**	**0,629**	0,956	0,977
SW_2	0.597	0.614	0.708	0.715
ME & CP_{δ_1}	0,590	0,591	0,978	0,979
PT & JWM	0,588	0,588	0,973	0,973
$WLCS$	0,587	0,590	0,974	0,977
ME & JWM	0,587	0,588	0,966	0,968
$WLCS$	0.587	0.590	0.974	0.977
PT & $WLCS$	0,586	0,588	0,982	0,984
ST & $WLCS$	0,587	0,585	**0,985**	**0,987**

Table 9: Top results for PatternBased-2. (CMS and CMS-2 stand for CombinedMostSimilar and CombinedMostSimilar-2 resp.)

Acc.	Alg.	m_1	m_2	k	score
PFN-1					
AA	CMS-2	ST & $WLCS$	ME & CP_{δ_2}	3	0,884
SR	CMS-2	$WLCS$	ME & CP_{δ_2}	3	0,887
AAR	CMS	ST & $WLCS$	ME & CP_{δ_2}	n.a.	0,979
RA	CMS	ST & $WLCS$	ME & CP_{δ_2}	n.a.	0,981
PFN-2					
AA	CMS-2	ME & JWM	SW_2	3	0,674
SR	CMS-2	ME & JWM	SW_2	3	0,715
AAR	CMS	ST & $WLCS$	PT & JWM	n.a.	0,987
RA	CMS-2	ME & CP_{δ_2}	ST & $WLCS$	2	0,988

Table 10: PatternBased-WithPreselection results

Dataset	AA	SR	AAR	RA
PFN-1	0,706	0,818	0,799	0,868
PFN-2	**0,660**	**0,775**	0,801	0,866

Finally, we have explored another very simple technique, which solely utilizes automatically acquired suffix-based patterns of the form $\{(f_{infl}, f_{base}), (l_{infl}, l_{base})\}$, where f_{infl} (l_{infl}), and f_{base} (l_{base}) stand for the corresponding suffixes in the inflected first name (surname) and base form of the first name (surname) resp. In other words, the inflection transitions for first names and surnames were not learned independently, but in parallel. The aforementioned patterns were extracted solely from the PFN-1 dataset. They were then used as follows. For an input name, all 'compatible' patterns were used in order to produce candidate base forms via performing appropriate suffix transitions of the first name and surname. If a candidate base form is in the search space, it is added to the results list. We will refer to this technique as ParallelPatterns. Application on PFN-1 resulted in following accuracy figures: AA - 82.0%, SR - 86.6%, AAR - 82.3%, and RA 86.8%. Clearly, they are not better than any top results obtained so far, but compared to PatternBased-WithPreselection they seem to perform better. In case of known order of first name and surname, this method constitutes an alternative, and should be studied more thoroughly, e.g., exploring usage of larger dictionary, which maps inflected forms to their base forms.

7. DISCUSSION

In section 6 we have presented results of numerous experiments on measuring lemmatization and name matching accuracy for several knowledge-poor methods. In particular, we have measured AA accuracy, which says how often a single-result answer constituting the base form could be returned (multiple-answer results are counted as false positives, i.e., they are penalized). Further, SR accuracy measures the precision of single-result answers w.r.t. returning a base form. Next, AAR accuracy measure gives the precision of returning the base form or some other variant of the same name, where multiple-answer are penalized again. Finally, the RA metric, the most 'relaxed' one, gives the percentage of results, which are either single-result answer or multiple-result answer, where all returned strings in the answer are either the base form or other variant of the same name.

In order to get a better picture of all results achieved with various techniques, an overview of the best AA, SR, AAR accuracy figures is given in Figures 6, 7, and 8 resp. The symbols **CP**, **S**, **R**, **CMS**, **CMS-2**, **PMS**, **PMS-2**, **PWP**, and **PP** correspond to CommonPrefix-MostSimilar, simple metrics, recursive metrics, CombinedMostSimilar algorithms, PatternBased algorithms, PatternBased WithPreselection, and ParallelPatterns method resp.

As can be observed, one can gain in AA accuracy via combining string distance metrics and further improve the accuracy figures by integrating automatically acquired suffix-based patterns for 'best' candidate selection. However, the integration of the latter ones results only in a small gain in accuracy. Interestingly, some fine-tuning of ParallelPatterns, e.g., via considering larger training dataset, would possibly result in accuracy gain for PFN-1. Nevertheless, going beyond the 90% mark seems to be difficult. In case of PFN-2 dataset, which contains harder to tackle cases (e.g, inversions, etc.) the AA accuracy figures are not very impressive, but this is due to the fact that in many cases the inverted base forms are being returned as the result (which is penalized). Most of the errors encounterd in the AA category were due to: (a) matching another variant of the same name, but not the base form itself, i.e., for many metrics distance between inflected variants is frequently smaller than between an inflected form and the corresponding base form, e.g., $dist$('$Ramazotiemu'$,'$Ramazotiego'$) is less than $dist$('$Ramazotiemu'$,'$Ramazoti'$), (b) reverse order of first name and surname in PFN-2, (c) homonymy of male and female variants of the same first name (see section 2), (d) similar surnames in the search space with wrong spelling, (e) inconsistency in declension of both first name and surname due to the declension rules for foreign first names and transliteration issues (see section 2). Interestingly, in Polish a base form of a proper name (masc) preserves original spelling while inflected versions use Polish transliteration.

As for AAR accuracy, similarly to AA, one could obtain best results for both datasets via combining string distance metrics and further significantly improve the accuracy by integrating automatically acquired suffix-based patterns. Due to the specification of AAR, PFN-1 and PFN-2 results were not much different except the simple metrics. Interestingly, almost optimal score could be achieved. Consequently, the best methods in the AAR category presented here are sufficient for performing person name matching tasks in Polish. Most likely, deployment of more sophisticated linguistic would not be highly beneficial.

The situation with SR is a bit different. The performance of almost all techniques, which go beyond the simple metrics is around 88-89%. Analogously to AA the figures for PFN-2 are not very impressive, but we could at least improve the SR figures via amalgamating various string distance metrics and other lightweight techniques. In the context of SR metric the top result obtained with CommonPrefix-MostSimilar should be ignored since it is probably due to the very small number of single-result answers, i.e., the other scores for this method were very poor, which indicates the low number of single-result answers.

As for RA figures, most of the top accuracy figures achieved with various methods for both datasets were oscillatting between 97% (simple metrics) and 98.7% (PatternBased algorithm). We do not discuss them in more detail here.

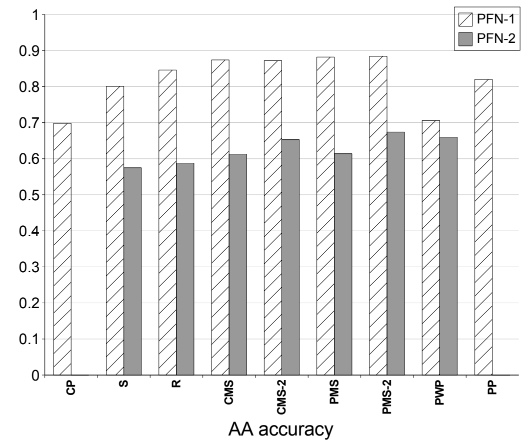

Figure 6: Summary of the AA accuracy

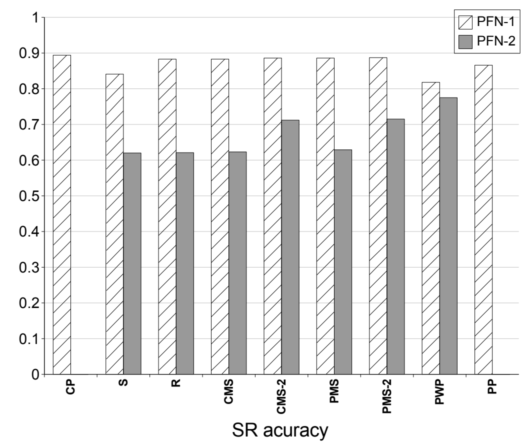

Figure 7: Summary of the SR accuracy

8. SUMMARY AND OUTLOOK

In this paper we studied the usability of several knowledge-poor methods for supporting and tackling the task of matching Polish person names. The presented techniques utilize string distance metrics, combinations thereof and automatically acquired suffix-based lemmatization patterns. The major aim of our work was to explore how good results can be obtained with such lightweight techniques without linguistic sophistication. For solving some of the tasks they seem to be suffcient, whereas for lemmatization, deployment of more elaborated techniques might result in better accuracy.

Since we did not consider and did not exploit the context the names appear in, the results presented in this paper constitute only useful guidelines for developing a fully-fledged solution to person name matching for Polish and similar highly inflective languages. To our knowledge this is one of the first efforts on tackling the person name matching task in Polish via application of linguistically poor methods.

Further, application of other machine learning techniques is envisaged too. For instance, in [17] a new probabilistic model for determining base forms for previously unseen

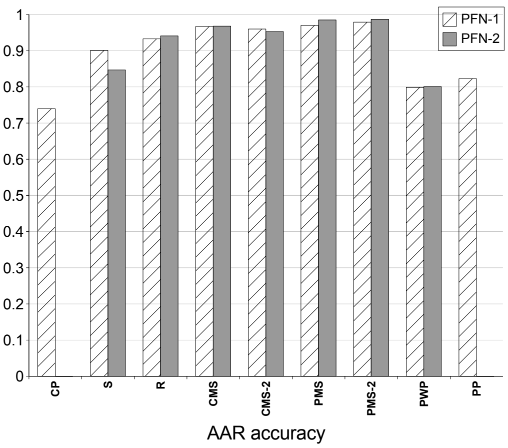

Figure 8: Summary of the AAR accuracy

words by analogy with a set word and base form pairs has been introduced. This new language-independent method for automatically learning a base form guesser, achieves a recall of 89-99% and precision of 76-94%, without any apriori knowledge of the declension paradigm. It would be interesting, to explore whether it could be utilized in the context of lemmatizing Polish person names.

Finally, we intend to apply the methods presented in this paper in a framework for clustering large web page collection in Polish according to persons mentioned in these pages.

To sum up, lemmatization of proper names and name matching in highly inflective languages poses an interesting and challenging problem. We strongly believe that work in this area is of paramount importance in the context of improving Web search quality since the number of non-English pages steadily increases.

9. ACKNOWLEDGEMENTS

The work reported in this paper was partially supported by the EMM project (`http://emm.jrc.it/overview.html`) carried out at the Joint Research Center of the European Commission. Further, the work was also supported by the Polish-Japanese Institute of Information Technology grants no. ST/SI/06/2006 and no. ST/SI/06/2007.

10. REFERENCES

[1] E. Agirre, L. Marquez, and R. Wicentowski. *Proceedings of SemEval2007 4th international Workshop on Semantic Evaluations, Prague, Czech Republic*. ACL, 2007.

[2] M. Bilenko and R. Mooney. Adaptive Duplicate Detection Using Learnable String Similarity Measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), Washington, USA*, 2003.

[3] P. Christen. A Comparison of Personal Name Matching: Techniques and Practical Issues. Technical report, TR-CS-06-02, Computer Science Laboratory, The Australian National University, Canberra, Australia, 2006.

[4] S. Coates-Steohens. The Analysis and Acquisition of Proper Names for the Understanding of a free Text. *Computers and the Humanities.*, 26:441–456, 1992.

[5] E. Cohen, P. Ravikumar, and S. Fienberg. A Comparison of String Metrics for Matching Names and Records. In *Proceedings of KDD Workshop on Data Cleaning and Object Consolidation*, 2003.

[6] W. Cohen, P. Ravikumar, and S. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78, Acapulco, Mexico, 2003.

[7] S. Cucerzan. Large Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the EMNLP-CoNLL Joint Conference, Prague, Czech Republic*. ACL, 2007.

[8] A. Elmagarmid, P. Ipeirotis, and V. Verykios. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1), 2007.

[9] I. Fellegi and A. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[10] L. Gravano, P. Ipeirotis, H. Jagadish, S. Koudas, N. Muthukrishnan, L. Pietarinen, and D. Srivastava. Using q-grams in a DBMS for Approximate String Processing. *IEEE Data Engineering Bulletin*, 24(4):28–34, 2001.

[11] J. Grzenia. *Słownik nazw własnych — ortografia, wymowa, słowotwórstwo i odmiana*. PWN, Warszawa, 1998.

[12] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA*, pages 127–138. ACM Press, 1995.

[13] H. Keskustalo, A. Pirkola, K. Visala, E. Leppanen, and K. Jarvelin. Non-adjacent digrams improve matching of cross-lingual spelling variants. In *Proceedings of SPIRE, LNCS 22857, Manaus, Brazil*, pages 252–265, 2003.

[14] A. Klementiev and D. Roth. Weakly Supervised Named-Entity Transliteration and Discovery from Multilingual Comparable Corpora. In *Proceedings of ACL 2006 Conference*. ACL, 2006.

[15] V. Levenshtein. Binary Codes for Correcting Deletions, Insertions, and Reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

[16] X. Li, P. Morie, and D. Rothd. Identification and Tracing of Ambiguous Names: Discriminative and Generative Approaches. In *Proceedings of the National Conference on Artificial Intelligence 2004*, 2004.

[17] K. Lindén. A Probabilistic Model for Guessing Base Forms of New Words by Analogy. In *Proceedings of CICling-2008, 9th International Conference on Intelligent Text Processing and Computational Linguistics, Haifa, Israel*, 2008.

[18] M. Miłkowski. Morfologik. Web document: *http://morfologik.blogspot.com*, 2007.

[19] A. Monge and C. Elkan. The Field Matching Problem: Algorithms and Applications. In *Proceedings of Knowledge Discovery and Data Mining 1996*, pages 267–270, 1996.

[20] B. On, D. Lee, J. Kang, and P. Mitra. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries, JCDL 2005, Denver, CA, USA*, pages 344–353. ACM, 2005.

[21] J. Piskorski. Named-Entity Recognition for Polish with SProUT. In L. Bolc, Z. Michalewicz, and T. Nishida, editors, *LNCS Vol 3490: Proceedings of IMTCI 2004, Warsaw, Poland.*, 2005.

[22] J. Piskorski, M. Sydow, and A. Kupść. Lemmatization of Polish Person Names. In *Proceedings of the ACL Workshop on Balto-Slavonic Natural Language Processing 2007 - Special Theme: Information Extraction and Enabling Technologies (BSNLP'2007). Held at ACL'2007, Prague, Czech Republic, 2007*. ACL Press, 2007.

[23] B. Pouliquen and R. Steinberger. Automatic Construction of Multilingual Name Dictionaries (in progress). *Learning Machine Translation*, 2008.

[24] T. Smith and M. Waterman. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[25] R. Steinberger and B. Pouliquen. Cross-lingual Named Entity Recognition. *Journal Linguisticae Investigationes, Special Issue on Named Entity Recognition and Categorisation*, 30(1):135–162, 2007.

[26] E. Ukkonen. Approximate String Matching with q-grams and Maximal Matches. *Theoretical Computer Science*, 92(1):191–211, 1992.

[27] D. Weiss. A Survey of Freely Available Polish Stemmers and Evaluation of Their Applicability in Information Retrieval. In *Proceedings of the 2nd Language and Technology Conference (LTC'2005), Poznań, Poland, 2005*, pages 216–221, 2005.

[28] D. Weiss. Korpus Rzeczpospolitej. Web document: *http://www.cs.put.poznan.pl/dweiss/rzeczpospolita*, 2007.

[29] W. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census, Washington, DC, 1999.