PROPERTIES AND ISSUES OF RULE-BASED NAMED ENTITY RECOGNITION

Miriam Kümmel - Winter Semester 2016/17

Universität Konstanz
Automated Media Content Analysis – PD Dr. Gerold Schneider

Structure

I. Introduction

Named entities are a major building block of “all textual data” (Mohamed, Oussalah 2014, p. 164) out of various classes. A named entity names a concept as a whole, without ascribing specific properties to it. It identifies an object within characterized classes (Didakowski 2006) and is usually a noun (Jungermann 2006, p. 1).

Named Entity Recognition therefore is a crucial sub-task of Information Extraction (henceforth IE) (Bajwa, Kaur 2015, p. 36). IE “turns the unstructured information embedded in texts into structured data” (Jurafsky, Martin 2009, p. 759). It is not sufficiently and officially defined what is included in IE, different researcher’s papers comprise varying tasks and concepts in IE (e.g. Jing Jiang: NER and Relation Extraction (see Charu C. Aggarwal, ChengXiang Zhai 2012); or Günter Neumann: among other things NER, Template Element Tasks, and Co-reference tasks (see PD Dr. Günter Neumann 2011)), but Named Entity Recognition indisputably has a great significance to all of them and researchers “noticed that it is essential to recognize [these] information units [=named entities]” (Nadeau, Sekine 2009, p. 3).

Named Entity Recognition is the “task of identifying sequences of terms in search queries that refer to a unique concept” (Eiselt, Figueroa 2013, p. 829). This unique concept can be referred to via different numeric or nominal mentions. A wider understanding of Named Entity Recognition also includes the recognition and classification of pronoun co-reference.

What expressions in a text constitute a named entity? A broad understanding of named entities would take in “names, including persons, organization and location names, and numeric expressions including time, date, money and percent expressions” (Nadeau, Sekine 2009, p. 3). Other, narrower, but commonly agreed on definitions of the term “named entity” only refer to “proper names” like the names of “’persons’, ‘locations’ and ‘organizations’” (Nadeau, Sekine 2009, p. 6; Mazur, Dale 2009, p. 51; Mohamed, Oussalah 2014, p. 164) whereby these categories can respectively be resolved into different sub-patterns like “city, state, country” (Nadeau, Sekine 2009, p. 6) for “location”.

This paper deals with the properties of named entities (What do they look like? How can they be spotted?) in the English language, the need for Named Entity Recognition (What is NER used for?) and the implementation and realisation of Named Entity Recognition (What are the different approaches to recognize and classify a NE?). Also, a primitive but robust Python code for Named Entity Recognition will be implemented and critically observed.

II. Theoretical Approaches to Named Entity Recognition

1. The linguistic features of named entities

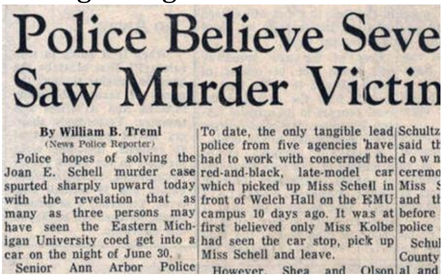
Even among linguistic experts, named entities (and proper names as a considerably large group among NEs) lack an “exhaustive description […] what would include a precise description of their linguistic properties”, so they can often not be classified or they are classified as “unknown words” (e.g. in e-dictionaries) (Vitas et al. 2009, p. 117). So, a characterization of named entities apart from their definition presented in the introduction can only be made “bottom up”: We look at named entities which we can spot with linguistic intuition and then gather their features.

A basic quality of named entities is that they form an open class: Every day new named entities are built (Didakowski 2006) (in high contrast to e.g. prepositions; even verbs and common nouns are not as productive) what makes the development of an adequate dictionary of all named entities in the world effectively impossible. Also, nearly every word in the common lexicon can be converted into a named entity (e.g. *Apple* (converting happened aorund 1976) or *(George W.) Bush* (converting happened a long time ago) (Didakowski 2006)). This also leads to a very high level of potential ambiguity concerning the classification of an expression (noun – named entity/proper noun) (Ekbal et al. 2009, p. 97). Another typical aspect of named entities is their heterogeneous appearance: There are one-word expressions like *Obama*, or longer ones across word boundaries like *Barack Obama*, *The New York Times* or *North Atlantic Treaty Organization*, acronyms like *NATO* or “regular” words like *(Condoleezza) Rice.*

But what qualities of an expression make a linguist think that this expression is a named entity?

1.1 Capitalisation

In German text data, “capitalisation is not a great help” (Cucerzan, Yarowsky 1999, p. 90) in spotting named entities as every noun is capitalised. Even though “English has a mixed and inconsistent [capitalisation] system which changes over time” (Moore 2000), named entities are one of the few capitalised word classes apart from the first word of a sentence and some exceptions like the pronoun *I*, words (not acronyms) all written in CAPS and some (newspaper) titles.



Police Believe Seve
Saw Murder Victin

By William B. Treml
(News Police Reporter)

Police hopes of solving the Joan E. Schell murder case spurted sharply upward today with the revelation that as many as three persons may have seen the Eastern Michigan University coed get into a car on the night of June 30.

Senior Ann Arbor Police

To date, the only tangible lead police from five agencies have had to work with concerned the red-and-black, late-model car which picked up Miss Schell in front of Welch Hall on the EMU campus 10 days ago. It was at first believed only Miss Kolbe had seen the car stop, pick up Miss Schell and leave.

However, Shea and Olson

Schultz
said th
d o w n
ceremo
Miss
and th
before
police
Schul
County
al and

Figure 1: Capitalised words in a newspaper title

1.2 Affixes

Affixes can be another morphological hint for an expression being a named entity. The Romanian suffix "-escu" for example is "an almost perfect indicator of a last name" while words beginning with "Mar-" very often are Romanian female first names (Cucerzan, Yarowsky 1999, p. 90). In English, morphological spotting of named entities is restricted to suffixes like "-son" or "-ez" (e.g. *Johnson*, *Anderson, Martinez*), which indicate some of the most frequent last names in the U.S. (Cucerzan, Yarowsky 1999, p. 90; Rhett Butler 2016). Many organization names also carry suffixes like "-ex", "-tech" or "-soft" (Nadeau, Sekine 2009, p. 12).

1.3 Inflection

Furthermore, named entities usually are not inflected[1]. Inflection differs a root of a word by the means of "place, time, and participant reference [e.g. person and number], without substantially affecting the basic semantic content of the root" (SIL International 2004). For instance, the static expression, i.e. named entity *The New York Times* is never used in a singular version (**The New York Time*). Derivational transfers, however, do occur: The Named Entity *Chopin* can for example be transferred into the adjective *Chopinian (tradition*) (Vitas et al. 2009, p. 118). Names can be inflected, too (e.g. *the Potters*).

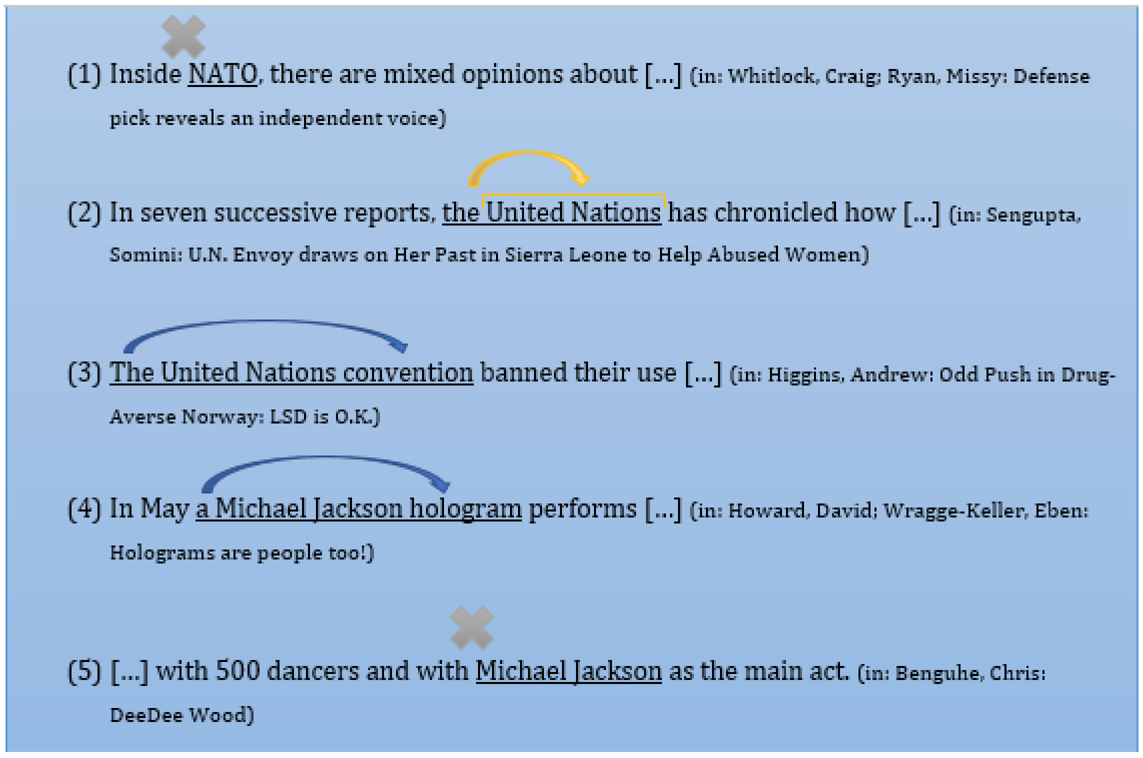
1.4 Surrounding

In many cases, named entities can also be spotted due to their surrounding syntactical patterns. An expression preceded by e.g. *Mr.* or *mayor of* mostly is a named entity (Jungermann 2006, p. 9).

1.5 Determiners

Named entities in English do not always, but sometimes have a determiner even though they are used as "normal" nominal expressions that would usually carry a determiner. The following examples express the inconsistency in the use of determiners with named entities[2]:

[1] This applies to the perspective on one language (in this case English). In the transmission to foreign languages, e.g. location names are often inflected: In Romanian *in Paris* becomes the inflected form *Parisului* (Steinberger, Pouliquen 2009, p. 148).
[2] Examples from Corpus of Contemporary American English: http://corpus.byu.edu/coca/



(1) Inside NATO, there are mixed opinions about [...] (in: Whitlock, Craig; Ryan, Missy: Defense pick reveals an independent voice)

(2) In seven successive reports, the United Nations has chronicled how [...] (in: Sengupta, Somini: U.N. Envoy draws on Her Past in Sierra Leone to Help Abused Women)

(3) The United Nations convention banned their use [...] (in: Higgins, Andrew: Odd Push in Drug-Averse Norway: LSD is O.K.)

(4) In May a Michael Jackson hologram performs [...] (in: Howard, David; Wragge-Keller, Eben: Holograms are people too!)

(5) [...] with 500 dancers and with Michael Jackson as the main act. (in: Benguhe, Chris: DeeDee Wood)

Figure 2: Usage of Determiners.

✖ *: no determiner;*

↷ *: determiner regarding noun compound;*

↷ *: determiner regarding Named Entity*

Apparently, the choice between using and not using a determiner is not only inconsistent (compare example (1) and (2)) but also depending on what follows the Named Entity. If it is followed by a noun, building a singular endocentric noun compound it is always preceded by a determiner (see example (3) and (4)) and some named entities are simply not preceded by a determiner (see example (5)). So unfortunately, (no) determiners are not a reliable indication of all the named entities but at least for names.

2. The need for Named Entity Recognition

Named Entity Recognition is a very basic part of natural language processing (henceforth NLP) on the level of morphology. But precisely for this reason, as so many further applications in NLP rely on the exactness of NER, its stability and reliability must be as high as possible. The following figure illustrates an exemplary development path of NLP implementations:

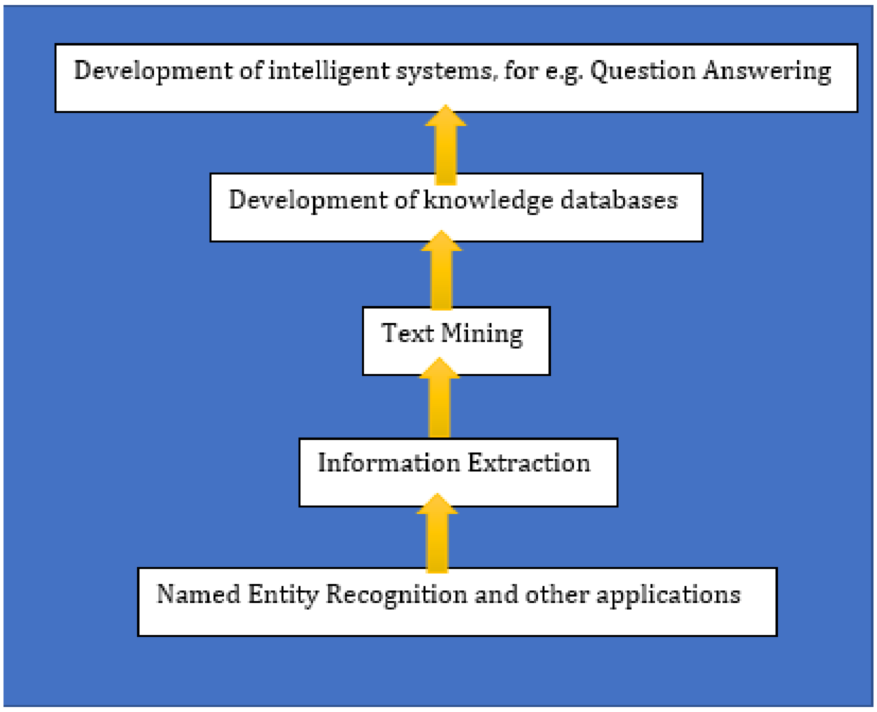
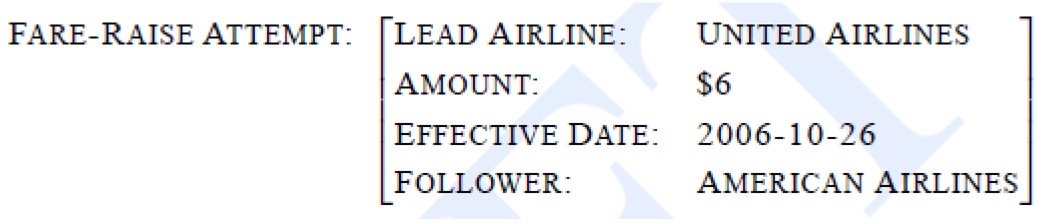

Development of intelligent systems, for e.g. Question Answering
Development of knowledge databases
Text Mining
Information Extraction
Named Entity Recognition and other applications

Figure 3: An exemplary development path for NLP implementations (Schneider, Zimmermann, p. 35; Galicia-Haro, Gelbukh 2009, p. 71)

Several NLP applications require "robust handling" (Galicia-Haro, Gelbukh 2009, p. 71) of named entities; the most prominent example is Information Extraction: "The starting point for most information extraction applications is the detection and classification of the named entities in a text" (Jurafsky, Martin 2009, p. 761). IE enables applications like e.g. search engines (Nadeau, Sekine 2009, p. 21) and automated template filling (Jurafsky, Martin 2009, p. 761).



FARE-RAISE ATTEMPT:	LEAD AIRLINE:	UNITED AIRLINES
	AMOUNT:	$6
	EFFECTIVE DATE:	2006-10-26
	FOLLOWER:	AMERICAN AIRLINES

Figure 4: Example for template filling (Jurafsky, Martin 2009, p. 761)

Additionally, NER can be described as an IE task itself, e.g. if someone is interested in all the mentions of named entities in a document (Jungermann 2006, p. 1). Further operations that are in need of NER are machine translation (Nadeau, Sekine 2009, p. 7), question answering, text classification and information retrieval (Galicia-Haro, Gelbukh 2009, p. 71).

In conclusion, it is not an exaggeration to say that some of the most popular applications on the web such as Google Translate, the Google search engine, Wolfram Alpha and IBM Watson rely on Named Entity Recognition.

3. Different implementing approaches – recognition and classification

Named Entity Recognition does not only achieve the recognition of named entities, it also categorizes them into several predetermined classes[3].

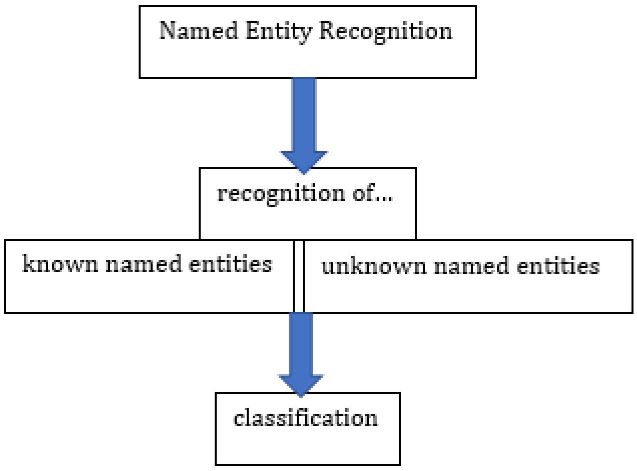


Figure 5: Phases in NER

In figure 5, the phases in NER are shown: The first step for the Named Entity Recognizer is to recognize an NE that is either known (e.g. *USA* (known for a very long time)) or unknown (e.g. the first time *Trump* occurring in a newspaper) (Galicia-Haro, Gelbukh 2009, p. 71) out of a given document (usually text). The second step in the process is the classification, also called grounding, of the NE. Grounding of an NE means that the NE is linked "to ontologies concepts" (sic!) (Batista 2011).

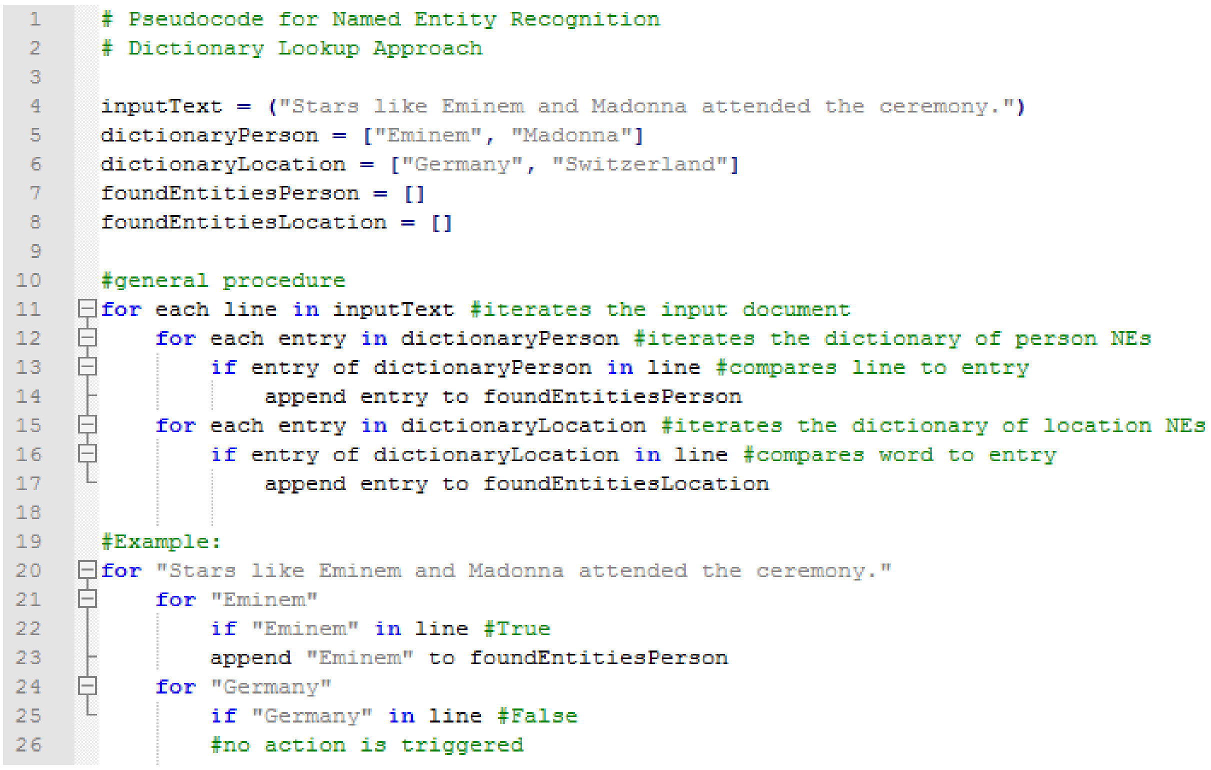
There are two main approaches to NER: rule-based and machine-learning-based (and hybrid) approaches (Liu et al. 2013, p. 1), whereas rules are the foundation for machine-learning-based NER systems, too. This paper will take a look at the rules that have to be implemented to recognize and classify as many NEs as possible. How effective can a tool be if it is solely rule-based? Which of the linguistic properties discussed in II.1. can help build a sensible NER? What do we miss if we do NER without machine learning algorithms and calculating properties?

[3] This is why e.g. Nadeau, Sekine 2009 refer to NER as NERC (Named Entity Recognition and Classification).

III. Rules for NER

1. The dictionary lookup – a basis

An uncomplicated and therefore robust approach to rule-based NER is the simple dictionary lookup. In this method, named entities are recognised "through a lookup procedure if a list of known [NE] exists" (Steinberger, Pouliquen 2009, p. 143). This implies that there is a list (in appropriate literature sometimes also called "gazetteer list" (Bajwa, Kaur 2015, p. 37)), i.e. dictionary, of NEs that can be looked up. For the recognition of NEs the implemented program iterates the given document and checks if it contains an NE that is listed in one of the dictionaries. If it is, an action like raising a counter or putting the found NE in an additional list is triggered. If an expression in the dictionaries is not found in the line, nothing happens. The aim of grounding can be achieved by creating several dictionaries with known NEs of different categories. The following basic pseudocode illustrates exemplarily how the dictionary lookup works.



```
# Pseudocode for Named Entity Recognition
# Dictionary Lookup Approach

inputText = ("Stars like Eminem and Madonna attended the ceremony.")
dictionaryPerson = ["Eminem", "Madonna"]
dictionaryLocation = ["Germany", "Switzerland"]
foundEntitiesPerson = []
foundEntitiesLocation = []

#general procedure
for each line in inputText #iterates the input document
    for each entry in dictionaryPerson #iterates the dictionary of person NEs
        if entry of dictionaryPerson in line #compares line to entry
            append entry to foundEntitiesPerson
    for each entry in dictionaryLocation #iterates the dictionary of location NEs
        if entry of dictionaryLocation in line #compares word to entry
            append entry to foundEntitiesLocation

#Example:
for "Stars like Eminem and Madonna attended the ceremony."
    for "Eminem"
        if "Eminem" in line #True
        append "Eminem" to foundEntitiesPerson
    for "Germany"
        if "Germany" in line #False
        #no action is triggered
```

Figure 6: Pseudocode for dictionary lookup

Although "lists are the privileged features" (Nadeau, Sekine 2009, p. 13) in NER, this approach obviously gathers some serious problems causing the accuracy of the program to drop:

(1) It violates the aim for recognizing unknown NE – even if the lists contain thousands of names, they will never cover every name existing (Galicia-Haro, Gelbukh 2009, p. 72).
(2) Especially when large documents are investigated with large dictionaries, the program will not be very efficient as every word in the document is compared with every entry of the dictionary.
(3) NEs cannot be disambiguated. If the NE “Rice” is in one of the dictionaries, the common noun “rice” will (if it is capitalised, e.g. at the beginning of a sentence) always be marked as an NE, even if it is used as a common noun.

So, apparently, this approach will work well regarding its robustness and reliability and can be used as a basis, but the issues mentioned above must be eradicated to achieve a sensible tool for NER.

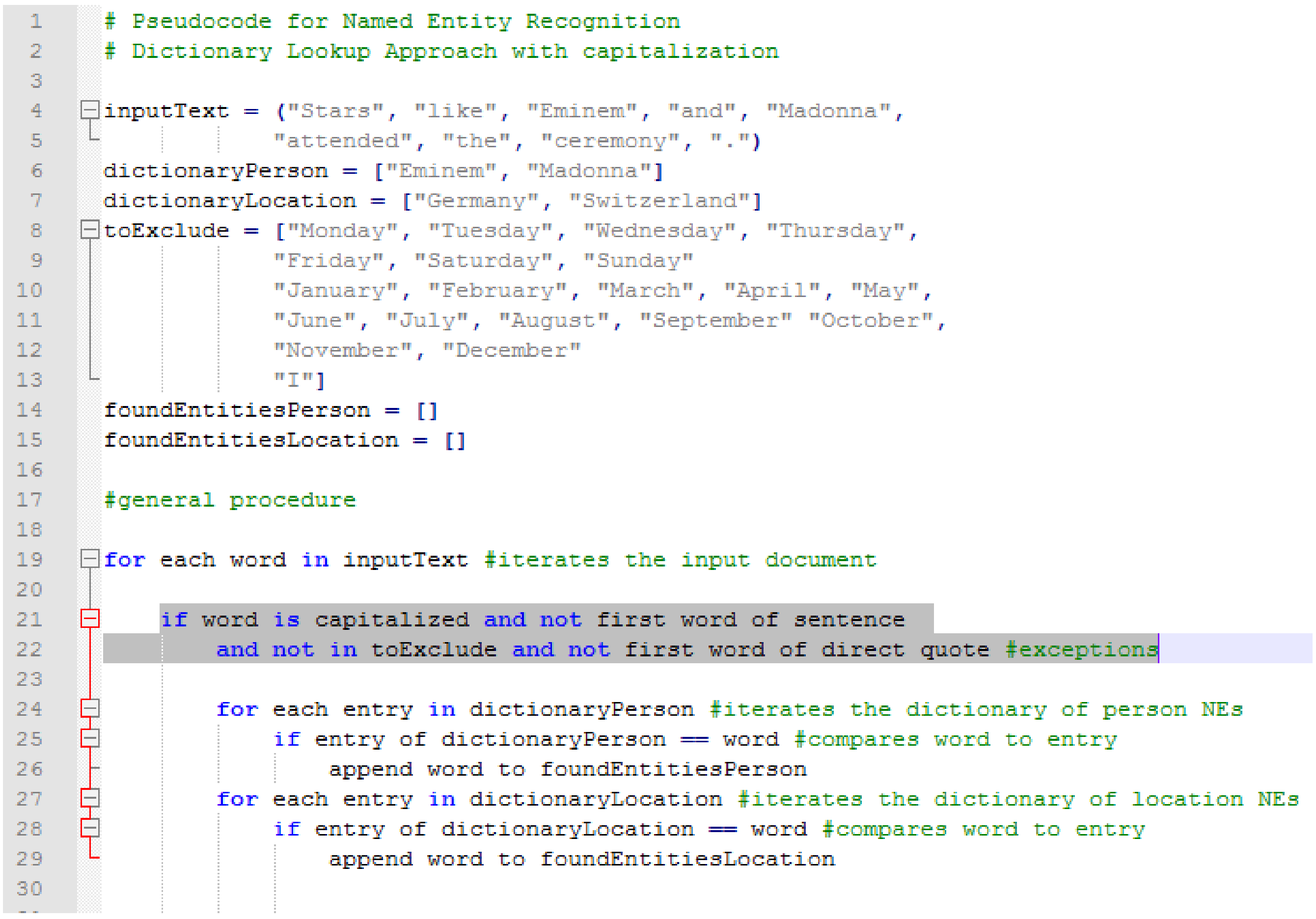
2. Capitalisation as a “sieve” – raising the efficiency

As mentioned in the theoretical discussion, NEs in English are capitalised most of the time[4]. They are not the only words capitalised but all in all there are only few expressions that have to be excluded:

- the first word of a sentence
- days of the week, months, holidays, seasons
- the first word in direct quotes
- the pronoun *I*

The dictionary lookup approach therefore could be improved regarding its efficiency by inserting a “sieve”: only capitalised words will be compared to the dictionaries while all the exceptions mentioned above must be considered. This change also requires a transition from iterating lines to iterating words (see line 19 in Figure 7) in this pseudocode to show the investigation of the document on the word-level.

[4] Of course, as always, there are exceptions (e.g. *eBay*).



```
1  # Pseudocode for Named Entity Recognition
2  # Dictionary Lookup Approach with capitalization
3
4  inputText = ("Stars", "like", "Eminem", "and", "Madonna",
5              "attended", "the", "ceremony", ".")
6  dictionaryPerson = ["Eminem", "Madonna"]
7  dictionaryLocation = ["Germany", "Switzerland"]
8  toExclude = ["Monday", "Tuesday", "Wednesday", "Thursday",
9              "Friday", "Saturday", "Sunday"
10             "January", "February", "March", "April", "May",
11             "June", "July", "August", "September" "October",
12             "November", "December"
13             "I"]
14 foundEntitiesPerson = []
15 foundEntitiesLocation = []
16
17 #general procedure
18
19 for each word in inputText #iterates the input document
20
21     if word is capitalized and not first word of sentence
22         and not in toExclude and not first word of direct quote #exceptions
23
24         for each entry in dictionaryPerson #iterates the dictionary of person NEs
25             if entry of dictionaryPerson == word #compares word to entry
26                 append word to foundEntitiesPerson
27         for each entry in dictionaryLocation #iterates the dictionary of location NEs
28             if entry of dictionaryLocation == word #compares word to entry
29                 append word to foundEntitiesLocation
30
```

Figure 7: Pseudocode for dictionary lookup with capitalisation as "sieve"

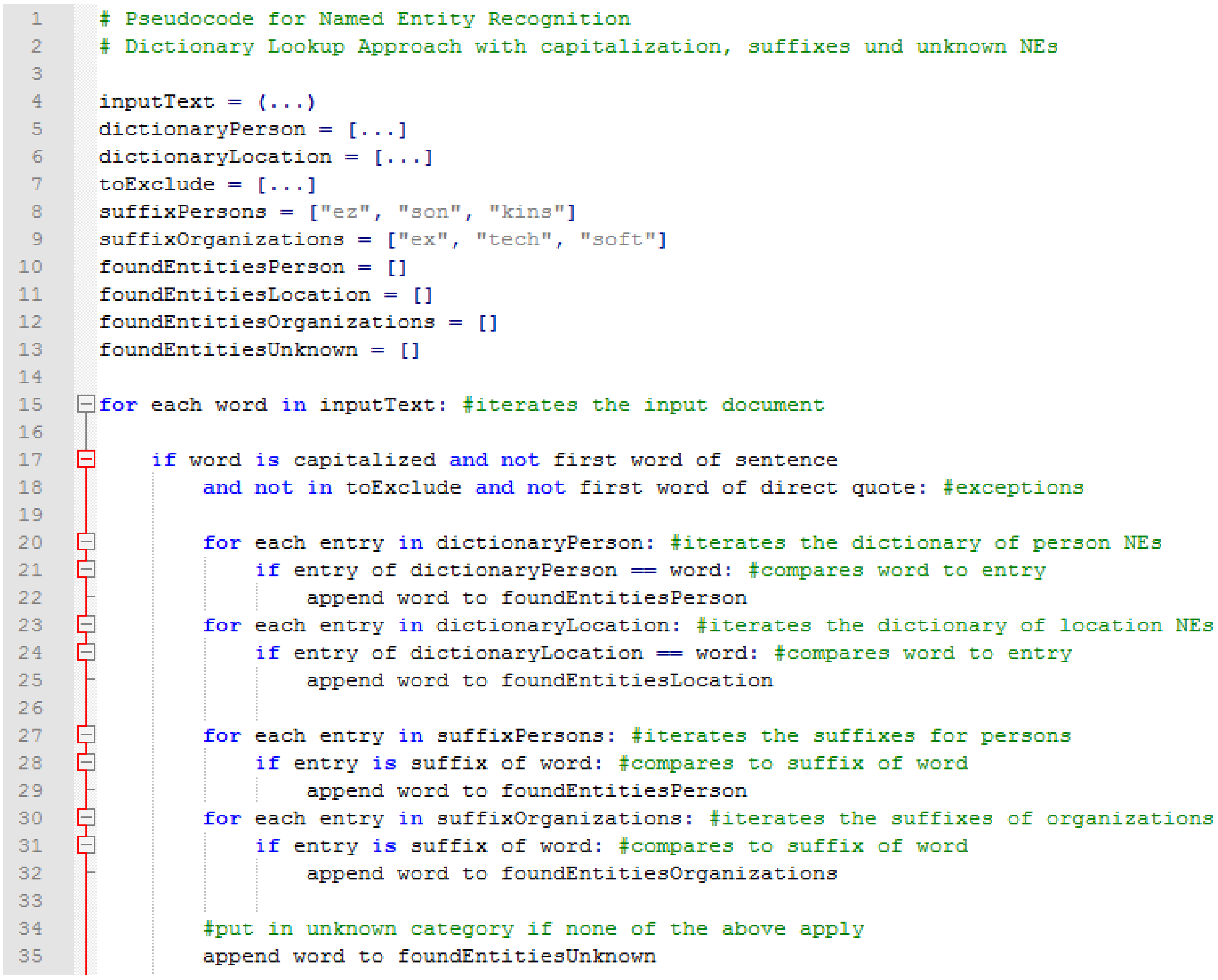
Marked in line 21 and 22, the "sieve" is applied: The searched word needs to be capitalised, not be the first word of a sentence or a direct quote and not be one of the words excluded[5]. This enhancement of the program helps to increase the efficiency of the tool in recognizing NEs by only investigating particular expressions, not every word.

It's quite possible that this tool will find NEs by applying its capitalisation rule that are not in one of the dictionaries. It would be important not to discard these NEs just because they are not part of a dictionary but store them in another reasonable place.

3. Considering suffixes – recognizing unknown NEs

After an NE is detected (because it is capitalised), the program checks if it is recorded in a dictionary. If not, its suffixes will be looked at as they often carry information about the affiliation of an NE (see line 27-32). If the suffix of the NE does not help with categorising it, the NE will be put in a list of NEs whose category is unknown (see line 35).

[5] This might not be a sufficient list of words to be excluded but further words can easily be added.



```
# Pseudocode for Named Entity Recognition
# Dictionary Lookup Approach with capitalization, suffixes und unknown NEs

inputText = (...)
dictionaryPerson = [...]
dictionaryLocation = [...]
toExclude = [...]
suffixPersons = ["ez", "son", "kins"]
suffixOrganizations = ["ex", "tech", "soft"]
foundEntitiesPerson = []
foundEntitiesLocation = []
foundEntitiesOrganizations = []
foundEntitiesUnknown = []

for each word in inputText: #iterates the input document

    if word is capitalized and not first word of sentence
        and not in toExclude and not first word of direct quote: #exceptions

        for each entry in dictionaryPerson: #iterates the dictionary of person NEs
            if entry of dictionaryPerson == word: #compares word to entry
                append word to foundEntitiesPerson
        for each entry in dictionaryLocation: #iterates the dictionary of location NEs
            if entry of dictionaryLocation == word: #compares word to entry
                append word to foundEntitiesLocation

        for each entry in suffixPersons: #iterates the suffixes for persons
            if entry is suffix of word: #compares to suffix of word
                append word to foundEntitiesPerson
        for each entry in suffixOrganizations: #iterates the suffixes of organizations
            if entry is suffix of word: #compares to suffix of word
                append word to foundEntitiesOrganizations

        #put in unknown category if none of the above apply
        append word to foundEntitiesUnknown
```

Figure 8: Pseudocode for dictionary lookup with capitalisation as "sieve" and consideration of suffixes and NEs of unknown category[6]

4. A small step towards disambiguation

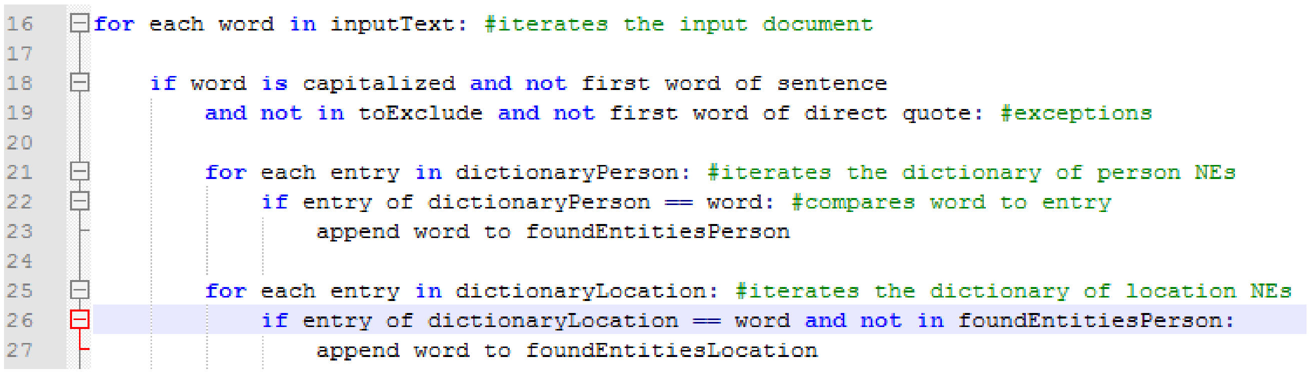
Word sense ambiguity affects NER in two ways: The recognition can be disturbed by ambiguity when the word is "in [an] ambiguous position (e.g., sentence beginning)" (Nadeau, Sekine 2009, p. 14). That way, the word *Brown* can either be an adjective in the beginning of a sentence or the last name of a person, i.e. a named entity. Therefore, it makes sense not to automatically treat expression in the beginning of a sentence as an NE (as they mostly are none) and to rather rely on the assumption that the expression (if it was a NE) will occur again, not in the beginning of a sentence.

A second problem affects the classification of named entities: Even if an NE is spotted correctly, its belonging to a class can be ambiguous, too. *Tony* for example can be a person

[6] Note that some lists are filled with [...] to save space and concentrate on the significant changes of the program.

name or one of the places (locations) in the USA called Tony (Steinberger, Pouliquen 2009, p. 148).

A small step towards avoiding false classifications is to “apply person name recognition first and to then block those words from being place name candidates” as “person name recognition is more reliable than location name recognition” (Steinberger, Pouliquen 2009, p. 149). A possible implementing solution is shown in the code-snippet in figure 9 (line 26): Name recognition is put before location recognition (just as before) and an entity can only be put in the dictionary of locations if it is not part of the dictionary of persons yet.



```
16  for each word in inputText: #iterates the input document
17
18      if word is capitalized and not first word of sentence
19          and not in toExclude and not first word of direct quote: #exceptions
20
21          for each entry in dictionaryPerson: #iterates the dictionary of person NEs
22              if entry of dictionaryPerson == word: #compares word to entry
23                  append word to foundEntitiesPerson
24
25          for each entry in dictionaryLocation: #iterates the dictionary of location NEs
26              if entry of dictionaryLocation == word and not in foundEntitiesPerson:
27                  append word to foundEntitiesLocation
```

Figure 9: Pseudocode-snippet for dictionary lookup with name recognition preference

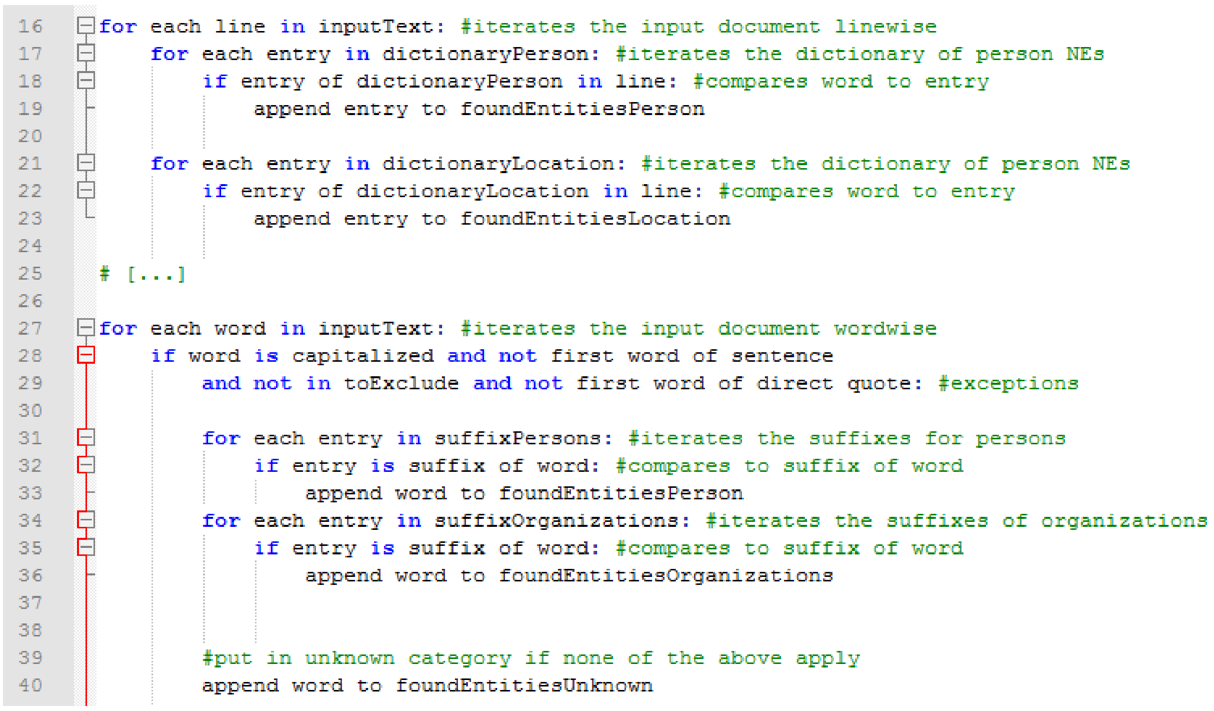
Following Nadeau et al. (2006), the recognition-problem (in their paper it is called “Entity-Noun-Ambiguity”) can be resolved by the ensuing procedure: An uppercase expression (e.g. *Jobs*, as the surname of Steve Jobs) will only be assumed an NE if it does not occur in lowercases in the same document. As soon as it “sometimes appears in the document without initial capitals” (i.e. *jobs*) it will not be accepted as an NE. In my opinion, this adjustment of the program is too coarse: It is conceivable that, for example as a rhetorical device, the NE-occurrence and the noun-occurrence of an expression both do appear in one document (e.g. *Steve Jobs created jobs.*[7]). For this reason, I will not adjust the code as proposed by Nadeau et al. (2006).

So far, a solely rule-based disambiguation of NEs without basing oneself on probabilistic methods or machine learning algorithms does not seem to be possible.

[7] See http://www.politifact.com/truth-o-meter/statements/2012/jan/26/mitch-daniels/mitch-daniels-says-steve-jobs-created-more-jobs-st/, checked on 02/27/2017

5. Re-Considering multiword expressions

Very often, NEs consist of more than one word (e.g. *The New York Times, Barack Obama, The Rolling Stones, …*). In the very beginning, multiword NEs were no problem for the code but in the meantime, we've changed to iterating single words instead of lines. For catching multiword NEs again, the document first is iterated line by line and afterwards word by word. This way, multiword NE expressions can be spotted and classified by the program before they are split up in line 27. All of the known NEs will be caught in the first part of the program (line 16-23) already. Only NEs that are not in a dictionary will get through to line 27-33 and be spotted and classified due to their suffixes. If none of the steps apply, the NE will after all be put in the list of unknown NEs.



```
16  for each line in inputText: #iterates the input document linewise
17      for each entry in dictionaryPerson: #iterates the dictionary of person NEs
18          if entry of dictionaryPerson in line: #compares word to entry
19              append entry to foundEntitiesPerson
20
21      for each entry in dictionaryLocation: #iterates the dictionary of person NEs
22          if entry of dictionaryLocation in line: #compares word to entry
23              append entry to foundEntitiesLocation
24
25  # [...]
26
27  for each word in inputText: #iterates the input document wordwise
28      if word is capitalized and not first word of sentence
29          and not in toExclude and not first word of direct quote: #exceptions
30
31          for each entry in suffixPersons: #iterates the suffixes for persons
32              if entry is suffix of word: #compares to suffix of word
33                  append word to foundEntitiesPerson
34          for each entry in suffixOrganizations: #iterates the suffixes of organizations
35              if entry is suffix of word: #compares to suffix of word
36                  append word to foundEntitiesOrganizations
37
38
39          #put in unknown category if none of the above apply
40          append word to foundEntitiesUnknown
```

Figure 10: Pseudocode-snippet for dictionary lookup with re-consideration of multiword expressions

IV. The scripted code

1. Description

The Python code can be found in the annex. Opened with Notepad++, the orange headings will facilitate the orientation for the reader.

The program follows the proceeding described in part III of the paper; the three following features were implemented:

(1) Dictionary lookup
(2) Capitalisation
(3) Suffix-research

The dictionary lookup did not pose a problem. I considered the order of the dictionaries (the persons-dictionary is iterated first for disambiguation aims) and made the program append the named entities only if they are not in the list yet to avoid duplicates. In the print-command there is a counter for the respective lists.

The usage of an RE seemed to be the best, or maybe even only, possibility to consider capitalisation in a document in Python. The RE has to be compiled first and can then be applied with different methods (I chose *.findall* to receive back a list). The syntax of REs is opaque and gets complex quite quickly. To make clear why I chose this exact RE, it is explained here.

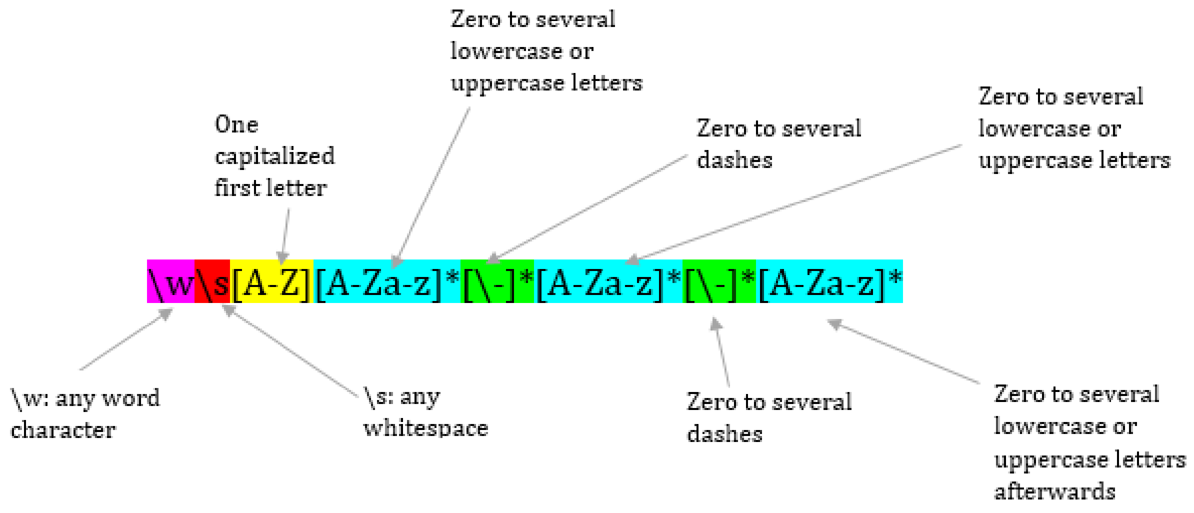

Zero to several lowercase or uppercase letters
One capitalized first letter
Zero to several dashes
Zero to several lowercase or uppercase letters
\w\s[A-Z][A-Za-z]*[\-]*[A-Za-z]*[\-]*[A-Za-z]*
\w: any word character
\s: any whitespace
Zero to several dashes
Zero to several lowercase or uppercase letters afterwards

Figure 11: Description of the applied RE

There might be a simpler way to achieve what this RE indicates but this way, even NEs like *You-Know-Who* can be recognised. The first two steps in the RE (\w\s) exclude all those capitalised words that are preceded by a punctuation mark and a whitespace or quotation marks thus some of the exceptions mentioned above are eliminated.

Excluding derivational or inflectional transfers and parts of the known NEs[8] was not as easy: the dictionaries had to be transformed into text and then be put in a list again, this time divided by whitespaces and then be iterated and compared to the unknown NEs.

The detection of the suffixes, again, was quite unproblematic as it strongly resembles the dictionary lookup implemented before.

The lists in this program are not complete, because if they were, one could not see how the program works. If the person names were all put in the dictionary for persons, there would be none left for the program to detect as "unknown NEs". Also, in investigating larger amounts of text one cannot simply adjust the dictionaries to the investigated data as this would push the necessity of an automatic program ad absurdum.

2. The downsides of the code

The processing of the program is displayed in the following figure.

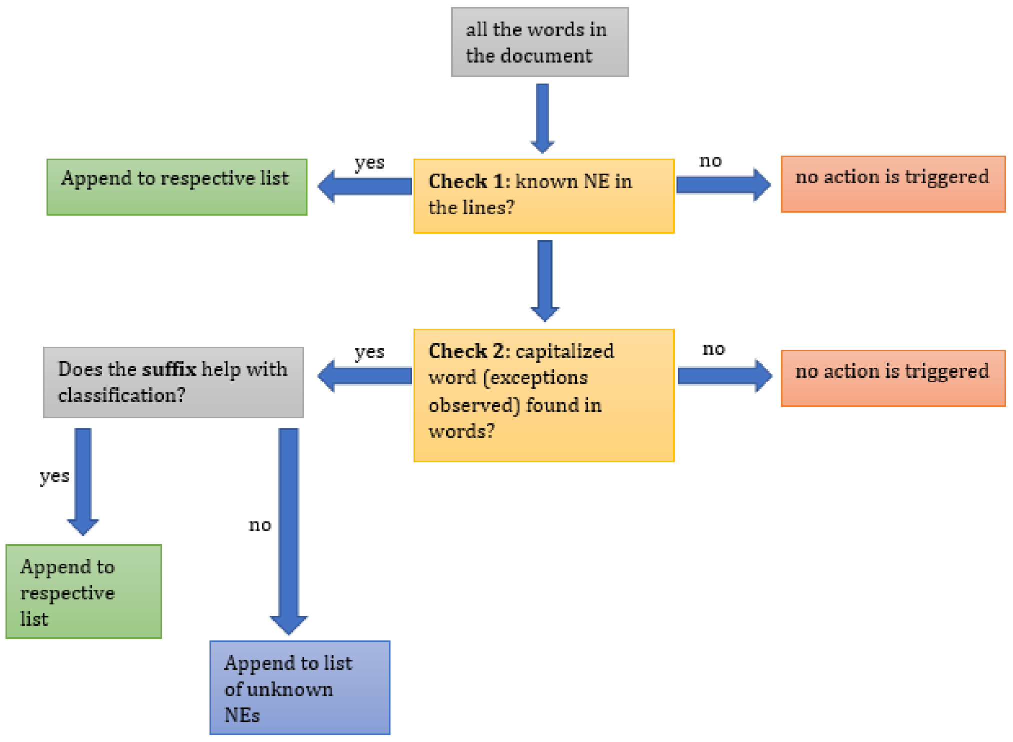

all the words in the document
Append to respective list
yes
Check 1: known NE in the lines?
no
no action is triggered
Does the suffix help with classification?
yes
Check 2: capitalized word (exceptions observed) found in words?
no
no action is triggered
yes
no
Append to respective list
Append to list of unknown NEs

Figure 12: The processing of the program

[8] i.e. not accept *Dursleys* as an unknown NE (*Dursley* is in the dictionary) and not accept *Privet* as an unknown NE (*Privet Drive* is in the dictionary)

It is becoming clear that the code strongly relies on the dictionaries, capitalisation and suffixes. Even though the code does work without any probabilities and machine learning algorithms, just as desired, one should note that some enormous simplifications are made here:

1. All the lists used in the program (lists of NEs, lists of suffixes, list of exceptions) are complete.
2. Capitalisation of NEs is observed by the author of the text and "normal" words (other than exceptions) are never capitalised.
3. Multiword NEs are not split up on line boundaries and pre-processing is done reliably.
4. There are no unknown lowercase NEs[9].
5. Expressions in the beginning of a sentence or just after a quotation mark are not (the only mention of) an NE.
6. There are no NEs that mix lowercase and uppercase expressions[10].

The recognition and especially the classification of unknown NEs is not satisfactory. Even if an NE that is not part of a dictionary is recognized (heavily relying on capitalisation) its classification is only made by some suffixes. Many NEs will not be classified at all. Also, unknown multiword NEs will not be recognized as a whole but always be split up and then be put in the list of unknown NEs.

This code is tailored to the investigated text (the beginning of *Harry Potter and the Sorcerer's Stone*). This means, that its accuracy will drop extremely if it is used on a different text. Sensible NER applications have to be customized towards their respective application domains (Yosef 2015, i) as there are so many domain-specific expressions. While the expression *Dursley* presumably never occurs in a text about biomedical advancement or even in the work of an author not being Joanne K. Rowling, it appeared more than 50 times in the text investigated here. This is one of the reasons why even a professional NER tool like the Stanford NER performs poorly on new genres like Tweets (Liu et al. 2013, p. 2).

In this code the surrounding of the NE has not been considered as it would involve further programming skills. But, examining and integrating the surrounding syntactical patterns of the NE can strongly boost the program's performance: This way, the program would recognize and classify even more NEs. For example, an expression preceded by *Dr.* or *Mr.* (person) or *the capital of* (location) in most cases is an NE. Also, "two adjacent capitalised

[9] The expression *unDursleyish* for example is not recognized by the program as it is not capitalised and not in a dictionary.
[10] E.g. the NE *Museum of Modern Art* would not be recognized as one NE as it mixes lowercase and uppercase expressions.

words in the middle of a text are likely to constitute a name" (Jurafsky, Martin 2009, p. 762) and several capitalised letters divided by dots (e.g. *U.S.*) tend to be an NE, too.

Of course, the developed approach shown here does not exploit the full potential of natural language processing at all but it is a robust access to Named Entity Recognition that is easy to understand and can be retraced even by programming beginners.

V. Conclusion and outlook

Named Entity Recognition is a crucial step in NLP. It is one of the few tasks that can be implemented only by rules and therefore does not forcingly rely on complex machine learning algorithms and probabilistic methods which can disturb the robustness of a tool and require large programming and implementing efforts. Also, a rule-based approach does not require costly annotated data. The outcome of a solely rule-based NER is not as accurate as it could be using machine learning and probabilities.

If an NER program was able to "learn" about named entities and their peculiarities, the success rate of the tool could be raised. There are various kinds of machine learning algorithms used for NER, e.g. pattern learning (the tool "learns domain-specific patterns" (Etzioni et al. 2004, p. 93), e.g. *to cities such as* is followed by a location NE) which is a special form of the clustering technique (NEs will be found based on the similarity of their context (Nadeau, Sekine 2009, p. 6)), or feature space vector models which represent documents in an abstract way of meaning (Nadeau, Sekine 2009, pp. 7–8). By this means, NER could for example easily be adapted to new text domains, which is not possible with solely rule-based tools.

In NER, machine learning algorithms usually are mixed with rule-based approaches (like gazetteer lists) and probabilistic methods (like Conditional Random Fields or Hidden Markov Model) to benefit from the advantages the respective methods provide (see Bajwa, Kaur 2015). With these hybrid approaches, best results can be achieved and one can benefit from high accuracy as well as from high stability and robustness.

But even the simple rule-based approach can be further developed. The classification of the NEs could be improved a lot: the shown tool only uses suffixes to classify the NEs. The consideration of the surrounding patterns as an improvement was mentioned already, but also, one could implement more affixes (e.g. the suffix *-ville* is a reliable indicator for a location NE) or consider all-caps expressions (e.g. *UNICEF*). Also, some NEs can be classified more detailed than as organization, location, person etc. Names can usually be divided into surnames and first names quite easily, also, female and male names can be clearly distinguished in most cases. This way, one could identify the NEs more precisely. As mentioned in the introduction, besides person, organization and location names, some researchers also consider "miscellaneous names" like "date, time, percentage and monetary expressions" (Ekbal et al. 2009, p. 97) as NEs which seem to be quite uncomplicated to implement in the tool (e.g. if digit is followed by *$*, append to list of monetary expressions).

As a further development, one could also consider titles: if there is a certain amount of capitalised words in a row, it may not be a single NE but a title.

After all, the reason why NER is such an elaborate task is the irregularity of named entities. Think of the title of the first Harry Potter novel: *Harry Potter and the Sorcerer's Stone*. It is a title, still, not every word is capitalised. Is *Sorcerer's Stone* capitalised because it is a NE or because the words are more important than *and the*? Who would think of an NE formed like *unDursleyish* when implementing a NER? The productiveness of the class of named entities will always make it difficult to recognize and classify them all.

VI. Annex

- Python-Code for Named Entity Recognition by Miriam Kümmel (PythonNERMiriamKuemmel.py)
- Extract of *Harry Potter and the Sorcerer's Stone* by Joanne K. Rowling (hp1.txt)

VII. Declaration of originality

I hereby declare:

I have composed this paper by myself and without any assistance other than the sources given in my list of works cited. This paper has not been submitted in the past or is currently being submitted to any other examination institution. It has not been published. All direct quotes as well as indirect quotes which in phrasing or original idea have been taken from a different text (written or otherwise) have been marked as such clearly and in each single instance under a precise specification of the source.

I am aware that any that any false claim made here results in failing the examination.

Nürnberg, 07.03.2017 [illegible]

VIII. List of figures

IX. Publication bibliography

Bajwa, Kanwalpreet Singh; Kaur, Amardeep (2015): Hybrid Approach for Named Entity Recognition (International Journal of Computer Applications (0975 – 8887), Volume 118 - No. 1).

Batista, David (2011): Named Entity Recognition and Grounding in News Articles. Data Management and Information Retrieval Group, REACTION workshop. Available online at http://dmir.inesc-id.pt/project-media/images/c/c8/Reaction-Talk-David-2011-04-14.pdf, checked on 2/23/2017.

Charu C. Aggarwal, ChengXiang Zhai (Ed.) (2012): Mining Text Data: Springer Science & Business Media.

Cucerzan, Silviu; Yarowsky, David (1999): Language Independent Named Entity Recognition Combining Morphological and Contextual Evidence.

Didakowski, Jörg (2006): Eigennamenerkennung mit großen lexikalischen Ressourcen. Tagung der Computerlinguistik Studierenden. Available online at http://www.coli.uni-saarland.de/conf/tacos-06/downloads/09_Eigennamenerkennung.pdf, checked on 2/21/2017.

Eiselt, Andreas; Figueroa, Alejandro (2013): A Two-Step Named Entity Recognizer for Open-Domain Search Queries. Conference Paper: Yahoo! Research Latin America.

Ekbal, Asif; Naskar, Sudip Kumar; Bandyopadhyay, Sivaji (2009): Named Entity Recognition and transliteration in Bengali. In Satoshi Sekine, Elisabete Ranchhod (Eds.): Named Entities. Recognition, classification and use: Benjamins Current Topics.

Etzioni, Oren; Cafarella, Michael; Downey, Doug; Popescu, Ana-Maria; Shaked, Tal; Soderland, Stephen et al. (2004): Unsupervised named-entity extraction from the Web: An experimental study. Washington.

Galicia-Haro, Sofía N.; Gelbukh, Alexander (2009): Complex named entities in Spanish texts. In Satoshi Sekine, Elisabete Ranchhod (Eds.): Named Entities. Recognition, classification and use: Benjamins Current Topics.

Jungermann, Felix (2006): Named Entity Recognition mit Conditional Random Fields. Dortmund.

Jurafsky, Dan; Martin, James H. (2009): Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition: Prentice Hall.

Liu, Xiaohua; Wei, Furu; Zhang, Shaodian; Zhou, Ming (2013): Named Entity Recognition for Tweets. In *ACM Transactions on Intelligent Systems and Technology*, 2013 (Vol. 4, No. 1, Article 3).

Mazur, Pawel; Dale, Robert (2009): Handling conjunctions in named entities. In Satoshi Sekine, Elisabete Ranchhod (Eds.): Named Entities. Recognition, classification and use: Benjamins Current Topics.

Mohamed, Muhidin; Oussalah, Mourad (2014): Identifying and Extracting Named Entities from Wikipedia Database using Entity Infoboxes.

Moore, Andrew (2000): Structure of English. Available online at http://www.universalteacher.org.uk/lang/engstruct.htm, checked on 2/22/2017.

Nadeau, David; Sekine, Satoshi (2009): A survey of named entity recognition and classification. In Satoshi Sekine, Elisabete Ranchhod (Eds.): Named Entities. Recognition, classification and use: Benjamins Current Topics.

Nadeau, David; Turney, Peter D.; Matwin, Stan (2006): Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity.

PD Dr. Günter Neumann (2011): Information Extraction. Architecture and Task Definition. Universität des Saarlandes, 2011. Available online at http://www.dfki.de/~neumann/InformationExtractionLecture2011/sessions/2-IE-Overview.pdf, checked on 2/21/2017.

Rhett Butler (2016): Most common last names in the United States, top 1000. Available online at http://names.mongabay.com/data/1000.html, checked on 2/23/2017.

Schneider, Gerold; Zimmermann, Heinrich: Text-Mining-Methoden im Semantic Web. In *Praxis der Wirtschaftsinformatik* 2010 (HMD 2).

SIL International (2004): What is inflection? Available online at http://www-01.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsInflection.htm, checked on 2/22/2017.

Steinberger, Ralf; Pouliquen, Bruno (2009): Cross-lingual Named Entity Recognition. In Satoshi Sekine, Elisabete Ranchhod (Eds.): Named Entities. Recognition, classification and use: Benjamins Current Topics.

Vitas, Duško; Krstev, Cvetana; Maurel, Denis (2009): A note on the semantic and morphological properties of proper names in the Prolex project. In Satoshi Sekine, Elisabete

Ranchhod (Eds.): Named Entities. Recognition, classification and use: Benjamins Current Topics.

Yosef, Mohamed Amir (2015): U-AIDA: a Customizable System for Named Entity Recognition, Classification, and Disambiguation: Saarbrücken.