Challenges in analyzing software documentation in Portuguese

Christoph TREUDE
Singapore Management University, ctreude@smu.edu.sg

Carlos A. PROLO

Fernando FIGUEIRA FILHO

Challenges in Analyzing Software Documentation in Portuguese

Christoph Treude, Carlos A. Prolo, Fernando Figueira Filho
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte
Natal, RN, Brazil
Email: {ctreude,prolo,fernando}@dimap.ufrn.br

***Abstract*—Many tools that automatically analyze, summarize, or transform software artifacts rely on natural language processing tooling for the interpretation of natural language text produced by software developers, such as documentation, code comments, commit messages, or bug reports. Processing natural language text produced by software developers is challenging because of unique characteristics not found in other texts, such as the presence of code terms and the systematic use of incomplete sentences. In addition, texts produced by Portuguese-speaking developers mix languages since many keywords and programming concepts are referred to by their English name. In this paper, we provide empirical insights into the challenges of analyzing software artifacts written in Portuguese. We analyzed 100 question titles from the Portuguese version of Stack Overflow with two Portuguese language tools and identified multiple problems which resulted in very few sentences being tagged completely correctly. Based on these results, we propose heuristics to improve the analysis of natural language text produced by software developers in Portuguese.**

***Keywords*—Documentation, natural language processing.**

I. Introduction and Motivation

In a typical software development process, developers perform several different activities: they use numerous tools to develop software artifacts ranging from source code and models to documentation and test scenarios; they use other tools to manage and coordinate their development work; and they spend a lot of time communicating with other members on their team [28]. In addition to source code, developers produce documents in natural language to disseminate knowledge inside and outside their development teams, including wikis [27], blogs [17], social media sites [15], the Question and Answer website Stack Overflow [16], or source code comments [22].

Several tools have been developed that automatically process natural language documents produced by software developers, for example by inferring specification from documentation [29], linking information from bug tracking systems and mailing lists to source code methods [14], summarizing bug reports [19], or extracting tasks from documentation [25]. Many of these tools rely on natural language processing tools such as the Stanford natural language processing toolkit [13] to split sentences, detect words in a sentence, assign parts of speech to words (such as *adjective*, *verb*, or *noun*), and to detect grammatical dependencies between different parts of a sentence (such as *subject* or *direct object*). Among these, part-of-speech tagging is particularly important since other parts of a natural language processing pipeline rely on it.

Previous work on analyzing natural language artifacts produced by software developers in English has developed heuristics such as automatically tagging all domain terms and source code terms as nouns [23], [25]. While these heuristics were able to produce satisfactory results, analyzing natural language artifacts produced by software developers is significantly more challenging when these artifacts are not in English, particularly because of the relatively lower quality of processing tools for other languages and the mix of languages caused by many technical terms still being referred to by their English name. As an example, in the question *"Qual é a diferença entre inner join e outer join?"* (What is the difference between inner join and outer join?), a tool would have to switch languages multiple times to detect *"inner"* as an English *adjective*, but *"e"* as a Portuguese *conjunction*.

To shed light on the particular challenges when analyzing natural language artifacts written by software developers in Portuguese, we downloaded the 100 most popular questions from the Portuguese version of Stack Overflow[1] and analyzed their titles with two Portuguese language tools. We then analyzed the part-of-speech tags that each of the tools assigned to the words in the question titles and annotated those that had been tagged incorrectly. Part-of-speech tagging is at the core of many software tools that use natural language processing to automatically analyze natural language text produced by software developers. For example, TaskNav, our previous work on extracting task descriptions from software documentation [26], would not be able to find meaningful tasks if the part-of-speech tags were set incorrectly.

Part-of-speech tagging is considered a relatively easy task in natural language processing with reported error rates of less than 3% in English for taggers such as Schmid's tree tagger [20]. As a highly inflected language, Portuguese should be expected to be even easier to tag since the great majority of the words are unambiguous due to the richness of morphological inflections. However, we found that only between 14 and 20% of the question titles were tagged fully correctly while the remaining 80 to 86% (depending on the tool) had at least one word tagged incorrectly. Out of a total of 974 words, the tools tagged between 161 (17%) and 182 (19%) incorrectly. We then categorized the different kinds of problems encountered by the tools and identified problems with words that were ambiguous, interrogative, infrequent, capitalized, or code-related. We propose several heuristics that have the potential to improve the part-of-speech tagging of software artifacts

[1]http://pt.stackoverflow.com/, snapshot taken on March 31, 2015.

written in Portuguese, in particular the establishment of a new part-of-speech tag *"Lexical Item"* for words that represent programming keywords, source code fragments, or technical terms in a different language.

II. Related Work

Work related to our research can be divided into work on analyzing natural language artifacts produced by software developers and work on the domain-specific analysis of natural language text, both in English and Portuguese. However, since very little previous work has specifically analyzed software documentation in Portuguese, our analysis of related work remains descriptive due to the lack of a baseline we could compare our work to.

A. Analyzing Software Artifacts

Several researchers have attempted to extract information from natural language artifacts produced by software developers. In an approach for inferring specifications from application programming interface (API) documentation, Zhong et al. used machine learning to detect actions and resources [29]. Panichella et al. developed an approach for automatically linking paragraphs from bug tracking systems and mailing lists to source code methods using a number of heuristics, such as the presence of the words *"call"*, *"execute"*, or *"invoke"* [14]. Petrosyan et al. proposed an approach for discovering tutorial sections that explain a given API type. They classified fragmented tutorial sections using supervised text classification based on linguistic and structural features [18].

In our previous work [25], [26], we developed a technique for automatically extracting tasks from software documentation by conceptualizing tasks as specific programming actions that have been described in the documentation. We used the grammatical dependencies between words in a sentence for our task extraction algorithm. More than 70% of the tasks we extracted were judged meaningful by at least one of two developers. A field study with professional developers using a search engine that presented the extracted tasks along with concepts, code terms, and section headers showed that search results identified through extracted tasks were more helpful to developers than other search results.

B. Domain-specific Analysis of English Text

A crucial challenge when applying natural language processing techniques to software artifacts, such as documentation, source code comments, or commit messages, is that these artifacts have unique characteristics not found in other natural language text. In a study comparing the effectiveness of six semantic similarity techniques on words from software comments and identifiers, Sridhara et al. found that customization of the similarity detection techniques was necessary to ensure a good performance [21]. Gupta et al. presented a part-of-speech tagger and syntactic chunker for source code names taking into account programmers' naming conventions, and they identified grammatical constructions that characterize a large number of program identifiers. Their approach led to a significant improvement of part-of-speech tagging of program identifiers [9]. In their description of a technique to "automate test automation" which uses as input a sequence of steps written in natural language and produces a sequence of procedure calls as output, Thummalapenta et al. maintained a repository of domain terms from the application under analysis and explicitly tagged those terms as single nouns [23]. We used a similar approach in our previous work [25], [26].

Domain adaptation of natural language processing has also received attention in areas other than software development. For example, Gimpel et al. proposed additional language processing features that leverage domain-specific properties of data from the micro-blogging service Twitter, such as orthography, frequently-capitalized words, and phonetic normalization. Their approach achieved almost 90% accuracy in tagging Twitter data [8].

C. Domain-specific Analysis of Portuguese Text

We also identified some relatively recent domain-specific work on Portuguese but none for the domain of software development. Among this work, Lopes et al. conducted an evaluation of compound term extraction from a corpus in the domain of Pediatrics by extracting bigrams and trigrams from a corpus of texts from the Portuguese journal *Journal de Pediatria* using three different extraction methods. The results of their analysis highlight the importance of verifying a threshold in the extraction process [12]. Lopes et al. also presented a tool [11] to extract relevant terms from Portuguese texts. Their tool extracts the most frequent noun phrases from an annotated corpus using the Palavras parser [2]. The tool allows for customizing the term extraction through linguistic and statistical criteria as well as for comparisons to manually constructed reference lists. Based on a pre-processed corpus of Portuguese texts, Ferreira et al. presented an automated method for the extraction of domain specific non-taxonomic relations [6]. To create domain-specific lexica, Fernandes et al. proposed a method that relies on the identification of unknown terms in a domain corpus. Their approach automatically detects unknown terms and identifies domain specific terms [4]. Hilgert et al. presented work on building domain-specific bilingual dictionaries based on a non-domain-specific method which they customized and parameterized. An evaluation on technical manuals in English and Portuguese with human specialists found significant increases in the precision of unigrams and multi-grams [10].

III. Methodology

In this section, we describe our methodology in terms of data collection and analysis.

A. Data Collection

To collect software artifacts produced in Portuguese, we accessed the Portuguese version of the Question and Answer website Stack Overflow, which facilitates the exchange of knowledge between programmers connected via the Internet. Answers on Stack Overflow often become a substitute for official product documentation when the official documentation is sparse or not yet existent [24]. As of April 2015, Stack Overflow contained almost 9.3 million questions, 15.5 million answers, and over 4 million users. The Portuguese version is much smaller, with almost 19 000 questions, 27 000 answers, and more than 15 000 users. We downloaded the 100 question

TABLE I. THE 10 MOST POPULAR SENTENCES ON THE PORTUGUESE VERSION OF STACK OVERFLOW AND CORRESPONDING PART-OF-SPEECH TAGS.

question title	**TreeTagger output**	**LX-Tagger output**
1 Como fazer hash de senhas de forma segura ?	**Como/*Conjunction*** fazer/*Verb* hash/*Noun* de/*Preposition* senhas/*Noun* de/*Preposition* forma/*Noun* segura/*Adjective* ?/*Punctuation*	**Como/*Conjunction*** fazer/*Verb* **hash/*Adjective*** de/*Preposition* senhas/*Noun* de/*Preposition* forma/*Noun* **segura/*Past Participle*** ?/*Punctuation*
2 As mensagens de erro devem se desculpar?	As/*Determiner* mensagens/*Noun* de/*Preposition* erro/*Noun* devem/*Verb* se/*Pronoun* desculpar/*Verb* ?/*Punctuation*	As/*Definite Article* mensagens/*Noun* de/*Preposition* erro/*Noun* devem/*Verb* se/*Clitic* desculpar/*Verb* ?/*Punctuation*
3 Como é feita a randomização pelo computador?	**Como/*Conjunction*** é/*Verb* feita/*Verb* **a/*Preposition*** **randomização/*Verb*** pelo/*Preposition+* computador/*Noun* ?/*Punctuation*	**Como/*Conjunction*** é/*Verb* feita/*Past Participle* a/*Definite Article* randomização/*Noun* por_/*Preposition* o/*Definite Article* computador/*Noun* ?/*Punctuation*
4 Por que devo usar apenas um ”return” em cada função ?	Por/*Preposition* que/*Interrogative* devo/*Verb* usar/*Verb* apenas/*Adverb* um/*Determiner* ”/*Quote* **return/Verb** ”/*Quote* em/*Preposition* cada/*Pronoun* **função/*Verb*** ?/*Punctuation*	Por/*Preposition* **que/*Relative*** devo/*Verb* usar/*Verb* apenas/*Adverb* um/*Indefinite Article* **”/*Noun*** **return/*Adjective*** ”/*Punctuation* em/*Preposition* cada/*Quantifier* função/*Noun* ?/*Punctuation*
5 Por que não devemos usar funções do tipo mysql_*?	Por/*Preposition* que/*Interrogative* **não/*Verb*** devemos/*Verb* usar/*Verb* **funções/*Verb*** do/*Preposition+Determiner* tipo/*Noun* mysql_*/*Noun* ?/*Punctuation*	Por/*Preposition* **que/*Relative*** não/*Adverb* devemos/*Verb* usar/*Verb* funções/*Noun* de_/*Preposition* o/*Definite Article* tipo/*Noun* **mysql/*Adjective* _/*Clitic* */*Verb*** ?/*Punctuation*
6 Qual é a diferença de API, biblioteca e Framework?	Qual/*Interrogative* é/*Verb* a/*Determiner* diferença/*Noun* de/*Preposition* API/*Noun* ,/*Punctuation* biblioteca/*Noun* e/*Conjunction* Framework/*Noun* ?/*Punctuation*	Qual/*Interrogative* é/*Verb* a/*Definite Article* diferença/*Noun* de/*Preposition* API/*Part of Name* ,/*Punctuation* biblioteca/*Noun* e/*Conjunction* Framework/*Part of Name* ?/*Punctuation*
7 Como fazer um algoritmo fonético para o português brasileiro?	**Como/*Conjunction*** fazer/*Verb* um/*Determiner* algoritmo/*Noun* **fonético/*Verb*** para/*Preposition* o/*Determiner* português/*Noun* brasileiro/*Adjective* ?/*Punctuation*	**Como/*Conjunction*** fazer/*Verb* um/*Indefinite Article* algoritmo/*Noun* fonético/*Adjective* para/*Preposition* o/*Definite Article* português/*Noun* brasileiro/*Adjective* ?/*Punctuation*
8 Diferenças entre Git, SVN e CVS	**Diferenças/*Verb*** entre/*Preposition* Git/*Noun* ,/*Punctuation* SVN/*Noun* e/*Conjunction* CVS/*Noun*	**Diferenças/*Part of Name*** entre/*Preposition* Git/*Part of Name* ,/*Punctuation* SVN/*Part of Name* e/*Conjunction* CVS/*Part of Name*
9 Como inverter uma string em javascript?	**Como/*Conjunction*** inverter/*Verb* uma/*Determiner* string/*Noun* em/*Preposition* javascript/*Noun* ?/*Punctuation*	**Como/*Conjunction*** inverter/*Verb* uma/*Indefinite Article* string/*Noun* em/*Preposition* javascript/*Noun* ?/*Punctuation*
10 Como prevenir injeção de código SQL no meu código PHP	**Como/*Conjunction*** prevenir/*Verb* **injeção/*Verb*** de/*Preposition* **código/*Verb*** SQL/*Noun* no/*Preposition+Determiner* meu/*Adjective* **código/*Verb*** PHP/*Noun*	**Como/*Conjunction*** prevenir/*Verb* injeção/*Noun* de/*Preposition* código/*Noun* SQL/*Part of Name* em_/*Preposition* o/*Definite Article* meu/*Possessive* código/*Noun* PHP/*Part of Name*

titles of the questions with the highest score on the Portuguese version of Stack Overflow on March 31, 2015. The left-most column of Table I shows the first ten of these question titles. We then analyzed each question title with two part-of-speech taggers for Portuguese: the tree tagger of Pablo Gamallo Otero [7][2] which is a parameterized version of Schmid's tree tagger [20] for Portuguese (referred to as TreeTagger in the remainder of this paper), and the shallow tagger of the LX-Suite [3][3] (LX-Tagger). We chose these two tools since they are available for free and because they represent two different streams of taggers.

B. Data Analysis

The first author annotated the output of both tools for each of the 100 question titles used as corpus in this study by indicating for each word whether it had been tagged correctly by the two tools. The correctness of this annotation was verified by the other two authors of this paper. The last two columns of Table I show the result for each of the ten most popular question titles where incorrect tags are indicated in bold. As can be seen from Table I, the tools use slightly different tag sets: The LX-Tagger has more specific part-of-speech tags and word features (*past participle*, *clitic*, *quantifier*, *part of name*, *possessive*, *definite / indefinite article*) while the TreeTagger is able to tag punctuation more precisely (e.g., *quote*). In addition, the tokenization (i.e., splitting a sentence into words) works slightly different. The LX-Tagger splits contractions (such as *"pelo"*) into two words (*"por"* and *"o"* in the example) and tags them separately. The same applies to code terms. For example, the LX-Tagger breaks *"mySQL_*"* into three words whereas the TreeTagger treats it as a single word. We took these particularities into account during our annotation and only annotated a tag as incorrect if there would have been a more appropriate tag available in the tag set of the specific tool.

[2]Available at http://gramatica.usc.es/~gamallo/php/tagger/TaggerPT.php.
[3]Available at http://lxcenter.di.fc.ul.pt/services/en/LXServicesSuite.html.

IV. RESULTS

TABLE II. TAGGING PROBLEMS.

	TreeTagger	**LX-Tagger**
failed to detect Code Term	44	35
failed to detect Interrogative	33	56
failed to detect Noun	62	21
failed to detect Punctuation	5	36
other	38	13
sum	182	161

Table II summarizes the results of our analysis. Out of a total of 974 words, the TreeTagger tagged 182 (19%) incorrectly, and the LX-Tagger tagged 161 (17%) incorrectly. In total, the TreeTagger resulted in only 14 question titles for which each word had been tagged correctly and the LX-Tagger resulted in 20 correctly tagged sentences. In the following sections, we discuss the different problems encountered by the taggers in more detail.

A. Detection of Code Terms

A unique characteristic of software-related natural language documents is the presence of code terms. Table III shows examples of where the taggers struggled with code terms interspersed with natural language text. In total, there were 44 cases within the 100 question titles where the TreeTagger tagged a code term incorrectly and 35 cases where the LX-Tagger had the same problem. For example, the LX-Tagger assigned a variation of part-of-speech tags to the JavaScript

TABLE III. DETECTION OF CODE TERMS.

1 Operador == e === em JavaScript *(LX-Tagger)* Operador/*Noun* **=/*Adjective* =/*Punctuation*** e/*Conjunction* **=/*Verb* =/*Past Participle* =/*Punctuation*** em/*Preposition* JavaScript/*Part of Name*
2 O que significa "??!??!" em linguagem C? *(TreeTagger)* O/*Determiner* que/*Interrogative* significa/*Verb* "/*Quote* **?/*Punctuation* ?/*Punctuation* !/*Punctuation* ?/*Punctuation* ?/*Punctuation* !/*Punctuation*** "/*Quote* em/*Preposition* linguagem/*Noun* C/*Noun* ?/*Punctuation*
3 Quando se deve usar var no javascript? *(TreeTagger)* **Quando/*Conjunction*** se/*Clitic* deve/*Verb* usar/*Verb* **var/*Infinitive*** em_/*Preposition* o/*Definite Article* javascript/*Noun* ?/*Punctuation*
4 Qual é a diferença entre inner join e outer join? *(LX-Tagger)* Qual/*Interrogative* é/*Verb* a/*Definite Article* diferença/*Noun* entre/*Preposition* **inner/*Verb*** join/*Noun* e/*Conjunction* **outer/*Verb*** join/*Noun* ?/*Punctuation*
5 Qual a diferença entre as funções var name = function() e function name()? *(LX-Tagger)* Qual/*Interrogative* a/*Definite Article* diferença/*Noun* entre/*Preposition* as/*Definite Article* funções/*Noun* **var/*Infinitive* name/*Noun* =/*Punctuation* function/*Interjection* (/*Part of Name*)/*Punctuation*** e/*Conjunction* **function/*Adjective* name/*Noun* (/*Adjective*)/*Punctuation*** ?/*Punctuation*

assignment operators == and ===: *adjective*, *punctuation*, *verb*, and *past participle*. Other examples include the treatment of the C operator `??!??!` as a series of *punctuation marks* by the TreeTagger and the treatment of the programming keyword `var` as an *infinitive* by the TreeTagger. It is debatable what the correct tag should be for operators such as == or +=, keywords such as `if` or `for`, and identifiers such as `var` or `main`. In our study, we considered it wrong when operators were not grouped and tagged as punctuation, and when keywords and identifiers were not tagged as nouns.

A particular challenge is the treatment of programming keywords that are also words in a natural language, such as *"inner"* and *"outer"* in the fourth example. To be able to assign the correct part-of-speech tags to the words in this question title, a tool would first have to realize that out of ten words in that question, four are in English. The last example shows a situation where a sentence does not only contain single code terms or keywords, but where entire code snippets are interspersed with the question text. In this case, the tagger assigned part-of-speech tags to the different parts of the code snippets instead of treating each snippet as a single word.

B. Detection of Interrogatives

TABLE IV. DETECTION OF INTERROGATIVES.

1 Como escrever um código legível e de fácil manutenção? *(TreeTagger)* **Como/*Conjunction*** escrever/*Verb* um/*Determiner* código/*Noun* legível/*Adjective* e/*Conjunction* de/*Preposition* **fácil/*Verb* manutenção/*Verb*** ?/*Punctuation*
2 Como funcionam closures em javascript? *(LX-Tagger)* **Como/*Adverb*** funcionam/*Verb* closures/*Noun* em/*Preposition* javascript/*Noun* ?/*Punctuation*
3 Onde devo colocar um código JavaScript em um documento HTML? *(TreeTagger)* **Onde/*Adverb*** devo/*Verb* colocar/*Verb* um/*Determiner* **código/*Adjective*** JavaScript/*Noun* em/*Preposition* um/*Determiner* documento/*Noun* HTML/*Noun* ?/*Punctuation*
4 O que é Reflection, por que é útil? *(LX-Tagger)* O/*Definite Article* **que/*Relative*** é/*Verb* Reflection/*Part of Name* ,/*Punctuation* por/*Preposition* **que/*Relative*** é/*Verb* útil/*Adjective* ?/*Punctuation*
5 O que realmente é DDD e quando ele se aplica? *(TreeTagger)* O/*Determiner* que/*Interrogative* realmente/*Adverb* é/*Verb* DDD/*Noun* e/*Conjunction* **quando/*Conjunction*** ele/*Pronoun* se/*Pronoun* aplica/*Verb* ?/*Punctuation*

The part-of-speech taggers we chose for this study had significant problems with detecting question words correctly: The TreeTagger mis-classified 33 interrogative words and the LX-Tagger mis-classified 56. Table IV shows examples of question words that were tagged as *conjunctions*, *adverbs*, or *relative pronouns*. Since this problem is not specific to software development – although questions are naturally very common on Question and Answer websites such as Stack Overflow – we do not analyze this issue further for now.

C. Detection of Nouns

TABLE V. DETECTION OF NOUNS.

1 Exceções consomem muito processamento. *(LX-Tagger)* **Exceções/*Part of Name*** consomem/*Verb* muito/*Adverb* processamento/*Noun* ./*Punctuation*
2 Quais são as técnicas para se fazer animações baseadas em scroll? *(TreeTagger)* Quais/*Interrogative* são/*Verb* as/*Determiner* **técnicas/*Adjective*** para/*Preposition* se/*Pronoun* fazer/*Verb* **animações/*Verb*** baseadas/*Verb* em/*Preposition* scroll/*Noun* ?/*Punctuation*
3 Requisição Ajax com Javascript puro (sem APIs) *(LX-Tagger)* **Requisição/*Part of Name*** Ajax/*Part of Name* com/*Preposition* Javascript/*Part of Name* puro/*Adjective* (/*Punctuation* sem/*Preposition* APIs/*Part of Name*)/*Punctuation*
4 Um compilador C pode gerar um executável de 64 bits onde ponteiros são 32-bits? *(TreeTagger)* Um/*Determiner* **compilador/*Adjective*** C/*Noun* pode/*Verb* gerar/*Verb* um/*Determiner* **executável/*Adjective*** de/*Preposition* 64/*Cardinal* bits/*Noun* **onde/*Interrogative*** ponteiros/*Noun* são/*Verb* 32-bits/*Noun* ?/*Punctuation*
5 Metodologias ágeis - um só programador *(LX-Tagger)* **Metodologias/*Part of Name*** ágeis/*Adjective* -/*Punctuation* um/*Indefinite Article* só/*Adjective* programador/*Noun*

In particular the TreeTagger failed to detect many software development related nouns, sometimes due to capitalization at the beginning of a sentence. The examples in Table V show that all of the following words were not detected as nouns by at least one of the tools in our study: *"exceções"* (exceptions), *"técnicas"* (techniques), *"animações"* (animations), *"requisição"* (request), *"compilador"* (compiler), *"executável"* (executable), and *"metodologias"* (methodologies). Instead, they were detected as *parts of a name*, *adjectives*, or *verbs*. All of these nouns are frequently used in natural language documents produced by software developers, and if a part-of-speech tagger cannot identify them as nouns, automated tools analyzing these documents will not function properly.

D. Detection of Punctuation

TABLE VI. DETECTION OF PUNCTUATION.

1 Dar um "SELECT" antes de um "INSERT" é uma forma segura de não ter registros duplicados? *(LX-Tagger)* **Dar/*Part of Name*** um/*Indefinite Article* **"/*Noun*** SELECT/*Part of Name* "/*Punctuation* antes/*Preposition* de/*Preposition* um/*Indefinite Article* **"/*Noun*** INSERT/*Part of Name* "/*Punctuation* é/*Verb* uma/*Indefinite Article* forma/*Noun* **segura/*Past Participle*** de/*Preposition* não/*Adverb* ter/*Verb* registros/*Noun* **duplicados/*Past Participle*** ?/*Punctuation*
2 Pode-se fazer o navegador "lembrar" de uma senha programaticamente? *(LX-Tagger)* Pode/*Verb* -se/*Clitic* fazer/*Verb* o/*Definite Article* navegador/*Noun* **"/*Verb*** lembrar/*Verb* "/*Punctuation* de/*Preposition* uma/*Indefinite Article* senha/*Noun* programaticamente/*Adverb* ?/*Punctuation*
3 Como funciona este if/else com "?" e ":"? *(LX-Tagger)* **Como/*Conjunction*** funciona/*Verb* este/*Demonstrative* if/*Noun* //*Symbol* else/*Noun* com/*Preposition* **"/*Noun*** ?/*Punctuation* "/*Punctuation* e/*Conjunction* "/*Part of Name* **:/*Part of Name* "/*Part of Name*** ?/*Punctuation*
4 Por que verde = azul + amarelo, mas no RGB amarelo = verde e vermelho? *(TreeTagger)* Por/*Preposition* que/*Interrogative* verde/*Adjective* **=/*Verb*** azul/*Adjective* **+/*Verb*** amarelo/*Adjective* ,/*Punctuation* mas/*Conjunction* no/*Preposition+Determiner* RGB/*Noun* amarelo/*Adjective* **=/*Verb*** verde/*Adjective* e/*Conjunction* vermelho/*Adjective* ?/*Punctuation*
5 Qual é a diferença semântica entre <em> e <strong>? *(LX-Tagger)* Qual/*Interrogative* é/*Verb* a/*Definite Article* diferença/*Noun* semântica/*Adjective* entre/*Preposition* <**em**> e/*Conjunction* <**strong**> ?/*Punctuation*

The correct tagging of punctuation turned out to be a problem particularly for the LX-Tagger. As the examples in Table VI show, quotation marks were tagged as *nouns*, *verbs*, and *parts of a name*. The same applies to the colon symbol (:). As the fourth example shows, the TreeTagger mis-classified the symbols in the equation *"verde = azul + amarelo"* (green = blue + yellow) as *verbs*. The last example shows that by default, the LX-Tagger ignores content enclosed in less-than (<) and greater-than (>) symbols and does not assign any part-of-speech tag. As a result, HTML tags such as `<em>` and `<strong>` in the example are not assigned any tag.

E. Other Issues

TABLE VII. OTHER ISSUES.

1 Utilizar muitas interfaces é uma má prática de programação? *(TreeTagger)* Utilizar/*Verb* muitas/*Adjective* interfaces/*Noun* é/*Verb* uma/*Determiner* **má/*Noun*** **prática/*Verb*** de/*Preposition* **programação/*Verb*** ?/*Punctuation*
2 Quais as implicações de não declarar variáveis em PHP? *(TreeTagger)* Quais/*Interrogative* **as/*Pronoun*** **implicações/*Verb*** de/*Preposition* **não/*Verb*** declarar/*Verb* **variáveis/*Verb*** em/*Preposition* PHP/*Noun* ?/*Punctuation*
3 E qual a sacada em usá-las? *(TreeTagger)* E/*Conjunction* qual/*Interrogative* a/*Determiner* sacada/*Noun* em/*Preposition* **usá-las/*Determiner*** ?/*Punctuation*
4 É correto dar maior preferência a composição do que herança? *(TreeTagger)* É/*Verb* correto/*Adjective* dar/*Verb* maior/*Adjective* **preferência/*Verb*** a/*Preposition* **composição/*Verb*** do/*Preposition+Determiner* **que/*Interrogative*** herança/*Noun* ?/*Punctuation*
5 Como -1 pode ser maior que 4? *(LX-Tagger)* **Como/*Adverb*** **-/*Punctuation*** 1/*Digit* pode/*Verb* ser/*Verb* maior/*Adjective* que/*Conjunction* 4/*Digit* ?/*Punctuation*

Table VII shows examples of other issues we found during our study but that occurred less frequently than the ones mentioned in the previous sections. These issues can usually be attributed to relatively complex sentence structures and include the mis-classification of *adjectives* (e.g., *"má"*), *articles* (e.g., *"as"*), *verbs* (e.g., *"usá-"*), *relative pronouns* (e.g., *"que"*), and *digits* (e.g., *"-1"*). It is important to note that one mis-classified word often negatively affects the tagging of other words in the same sentence. For example, in the second example sentence, the TreeTagger failed to tag *"implicações"* (implications) as a *noun* which led to the mis-classification of the preceding *definite article* (*"as"*). To prevent such issues, in the next section, we present our recommendations for improving the part-of-speech tagging of software documents produced by software developers in Portuguese.

V. RECOMMENDATIONS

We make four recommendations that have the potential to improve the part-of-speech tagging discussed in the previous sections.

A. Introducing a Lexical Item tag

Our first recommendation is the introduction of a new part-of-speech tag which we call *"Lexical Item"*. The need for such a tag is motivated by several scenarios where we were unable to find a fitting tag in the current tag sets for a particular word. For example, it is impossible to assign the "correct" part-of-speech tag to the word *"if"* in the question title *"Por que em algumas situações if's são considerados ruins?"* (Why are if's considered bad in some situations?). The word *"if"* in the English language is used as a *subordinating conjunction* which can never have a plural form. Given the sentence structure in the Portuguese question title, the word *"if"* should actually be treated as a *noun*, with the plural *"if's"*. Since neither solution makes sense and current tools struggle to assign a part-of-speech tag in such situations (the TreeTagger tagged *"if"* as a *verb* and the LX-Tagger tagged it as an *adjective*), we propose to tag the word instead as a *"Lexical Item"*. This tag would be used for code snippets (such as *"var name = function()"*), programming keywords (such as *"if"*), and technical terms borrowed from another language (such as *"inner join"*).

B. Dictionary of Programming Keywords

Since programming keywords may or may not be words in a natural language (for example, `if` is an English word whereas `int` is not), a part-of-speech tagger would benefit from a dictionary of programming keywords. Such a dictionary could help identify keywords as *"Lexical Items"* and help with the detection of code snippets (see next section). We found that tools struggle with finding appropriate part-of-speech tags for programming keywords. For example, `return` was tagged as *adjective* by the LX-Tagger while `heap` and `lib` were tagged as *verbs* by the TreeTagger. A multilingual dictionary [5] that contains Portuguese words as well as common programming keywords could avoid such problems.

C. Regular Expressions to Detect Code Terms

Several researchers have developed regular expressions for identifying code snippets and code terms in natural language text [1], [25]. We propose to combine these approaches with part-of-speech tagging of natural language documents produced by software developers in Portuguese. In particular in scenarios where a sentence contains entire code snippets, such as in the question title *"Qual a diferença entre as funções var name = function() e function name()?"* (What is the difference between the functions var name = function() and function name()?), regular expressions combined with programming keywords could be used to indicate where the Portuguese text ends and the source code starts. Entire code snippets could then be grouped and tagged with a single part-of-speech tag (i.e., the suggested *"Lexical Item"* tag).

D. Software-related Nouns

Both taggers struggled to correctly identify software-related nouns, such as *"animações"* (animations), which was tagged as a *verb* by the TreeTagger. Even though a Portuguese word was used, it appears that part-of-speech tagging tools are not trained on words that are rarely used outside of software development. To address this concern, we propose to support Portuguese part-of-speech taggers by providing a list of words that are commonly used as nouns in natural language artifacts produced by software developers.

VI. CONCLUSION AND FUTURE WORK

Tools that automatically analyze software artifacts written in natural language, such as documentation, code comments, or bug reports, rely on natural language processing tools for the interpretation of text. Processing text produced by software developers is challenging because of unique characteristics not found in other texts, such as the presence of code terms and the use of incomplete sentences. Artifacts produced in

Portuguese have the additional challenge of mixing Portuguese text with English words for many programming concepts. While we cannot claim that our results generalize beyond the top 100 question titles from the Portuguese version of the Question and Answer website Stack Overflow, the results are a strong indication that part-of-speech tagging of software artifacts written in Portuguese is challenging and that current tools are unable to achieve reasonable results. We make several recommendations for addressing this situation, ranging from the introduction of a *"Lexical Item"* part-of-speech tag to the use of dictionaries and regular expressions for the detection of parts of a sentence that are not standard Portuguese.

In future work, we plan to deepen our analysis by considering other natural language artifacts produced by software developers in Portuguese, such as bug reports and tutorials. We will also implement the heuristics we recommend and evaluate their performance on the data set used in this paper as well as on other artifacts from the growing population of natural language documents and annotations produced by software developers, both in English and Portuguese. Increasing the accuracy of part-of-speech tagging of natural language texts will enable researchers and practitioners to automatically analyze these texts, ultimately leading to better artifacts and more efficient development processes.

Acknowledgements

This work is partially supported by CNPq Jovens Talentos grant 407455/2013-2, CNPq Universal grant 460904/2014-0, and CAPES/PROAP.

References

[1] A. Bacchelli, M. D'Ambros, and M. Lanza. Extracting source code from e-mails. In *18th Int'l. Conf. on Programming Comprehension*, pages 24–33, 2010.

[2] E. Bick. *The parsing system "Palavras": Automatic grammatical analysis of Portuguese in a constraint grammar framework*. Aarhus Universitetsforlag, 2000.

[3] A. Branco and J. R. Silva. A suite of shallow processing tools for Portuguese: LX-Suite. In *Proc. of the 11th Conf. of the European Chapter of the Association for Comp. Linguistics: Posters & Demonstrations*, pages 179–182, 2006.

[4] P. Fernandes, L. O. Furquim, and L. Lopes. A supervised method to enhance vocabulary with the creation of domain specific lexica. In *Proc. of the Int'l. Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, pages 139–142, 2013.

[5] P. Fernandes, L. Lopes, C. A. Prolo, A. Sales, and R. Vieira. A fast, memory efficient, scalable and multilingual dictionary retriever. In *Proc. of the Lang. Resources and Evaluation Conf.*, pages 2520–2524, 2012.

[6] V. H. Ferreira, L. Lopes, R. Vieira, and M. J. Finatto. Automatic extraction of domain specific non-taxonomic relations from Portuguese corpora. In *Proc. of the Int'l. Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, pages 135–138, 2013.

[7] P. Gamallo Otero and I. González López. A grammatical formalism based on patterns of part-of-speech tags. *Int'l. Journal of Corpus Linguistics*, 16(1):45–71, 2011.

[8] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for Twitter: Annotation, features, and experiments. In *Proc. of the 49th Annual Meeting of the Association for Comp. Linguistics: Human Lang. Technologies: short papers - Volume 2*, pages 42–47, 2011.

[9] S. Gupta, S. Malik, L. Pollock, and K. Vijay-Shanker. Part-of-speech tagging of program identifiers for improved text-based software engineering tools. In *Proc. of the 21st Int'l. Conf. on Programming Comprehension*, pages 3–12, 2013.

[10] L. Hilgert, L. Lopes, A. Freitas, R. Vieira, D. Hogetop, and A. Vanin. Building domain specific bilingual dictionaries. In *Proc. of the 9th Int'l. Conf. on Lang. Resources and Evaluation*, pages 2772–2777, 2014.

[11] L. Lopes, P. Fernandes, R. Vieira, G. Fedrizzi, and D. Martins. EχATOLp–a tool for domain relevant terms extraction. In *Proc. of the Int'l. Conf. on Comp. Processing of the Portuguese Lang.*, 2010.

[12] L. Lopes, R. Vieira, M. J. Finatto, and D. Martins. Extracting compound terms from domain corpora. *Journal of the Brazilian Computer Society*, 16(4):247–259, 2010.

[13] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proc. of 52nd Annual Meeting of the Association for Comp. Linguistics: System Demonstrations*, pages 55–60, 2014.

[14] S. Panichella, J. Aponte, M. D. Penta, A. Marcus, and G. Canfora. Mining source code descriptions from developer communications. In *Proc. of the 20th Int'l. Conf. on Programming Comprehension*, pages 63–72, 2012.

[15] C. Parnin and C. Treude. Measuring API documentation on the web. In *Proc. of the 2nd Int'l. workshop on Web 2.0 for Software Engineering*, pages 25–30, 2011.

[16] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey. Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow. Technical Report GIT-CS-12-05, Georgia Institute of Technology, 2012.

[17] C. Parnin, C. Treude, and M.-A. Storey. Blogging developer knowledge: Motivations, challenges, and future directions. In *Proc. of the 21st Int'l. Conf. on Programming Comprehension*, pages 211–214, 2013.

[18] G. Petrosyan, M. P. Robillard, and R. de Mori. Discovering information explaining API types using text classification. In *Proc. of the 37th Int'l. Conf. on Software Engineering*, 2015. To appear.

[19] S. Rastkar, G. C. Murphy, and G. Murray. Automatic summarization of bug reports. *IEEE Trans. on Software Engineering*, 40(4):366–380, 2014.

[20] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proc. of the Int'l. Conf. on New Methods in Lang. Processing*, volume 12, pages 44–49, 1994.

[21] G. Sridhara, E. Hill, L. Pollock, and K. Vijay-Shanker. Identifying word relations in software: A comparative study of semantic similarity tools. In *Proc. of the 16th Int'l. Conf. on Programming Comprehension*, pages 123–132, 2008.

[22] M.-A. Storey, J. Ryall, J. Singer, D. Myers, L.-T. Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE Trans. on Software Engineering*, 35(4):470–483, 2009.

[23] S. Thummalapenta, S. Sinha, D. Mukherjee, and S. Chandra. Automating test automation. Technical Report RI11014, IBM Research, 2011.

[24] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web? (NIER track). In *Proc. of the 33rd Int'l. Conf. on Software Engineering*, pages 804–807, 2011.

[25] C. Treude, M. P. Robillard, and B. Dagenais. Extracting development tasks to navigate software documentation. *IEEE Trans. on Software Engineering*, 41(6):565–581, 2015.

[26] C. Treude, M. Sicard, M. Klocke, and M. P. Robillard. TaskNav: Task-based navigation of software documentation. In *Proc. of the 37th Int'l. Conf. on Software Engineering*, 2015. To appear.

[27] C. Treude and M.-A. Storey. Effective communication of software development knowledge through community portals. In *Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering*, pages 91–101, 2011.

[28] C. Treude and M.-A. Storey. Work item tagging: Communicating concerns in collaborative software development. *IEEE Trans. on Software Engineering*, 38(1):19–34, 2012.

[29] H. Zhong, L. Zhang, T. Xie, and H. Mei. Inferring resource specifications from natural language API documentation. In *Proc. of the 24th Int'l. Conf. on Automated Software Engineering*, pages 307–318, 2009.