Expert Systems with Applications 41 (2014) 853–860

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Syntactic N-grams as machine learning features for natural language processing

Grigori Sidorov [a,*], Francisco Velasquez [a], Efstathios Stamatatos [b], Alexander Gelbukh [a], Liliana Chanona-Hernández [c]

[a] Center for Computing Research (CIC), Instituto Politécnico Nacional (IPN), Mexico City, Mexico
[b] University of the Aegean, Greece
[c] ESIME, Instituto Politécnico Nacional (IPN), Mexico City, Mexico

ARTICLE INFO

Keywords:
Syntactic n-grams
sn-Grams
Parsing
Classification features
Syntactic paths
Authorship attribution
SVM
NB
J48

ABSTRACT

In this paper we introduce and discuss a concept of syntactic n-grams (sn-grams). Sn-grams differ from traditional n-grams in the manner how we construct them, i.e., what elements are considered neighbors. In case of sn-grams, the neighbors are taken by following syntactic relations in syntactic trees, and not by taking words as they appear in a text, i.e., sn-grams are constructed by following paths in syntactic trees. In this manner, sn-grams allow bringing syntactic knowledge into machine learning methods; still, previous parsing is necessary for their construction. Sn-grams can be applied in any natural language processing (NLP) task where traditional n-grams are used. We describe how sn-grams were applied to authorship attribution. We used as baseline traditional n-grams of words, part of speech (POS) tags and characters; three classifiers were applied: support vector machines (SVM), naive Bayes (NB), and tree classifier J48. Sn-grams give better results with SVM classifier.

1. Introduction

N-gram based techniques are predominant in modern natural language processing (NLP) and its applications. Usually, they are used as features in representing vector space model and then the standard classification algorithms are applied for this model. The values of these features are frequencies of the n-grams, possibly weighted in some manner.

Traditional n-grams are sequences of elements as they appear in texts. These elements can be words, characters, POS tags,[1] or any other elements as they appear one after another in texts. Common convention is that "n" in n-grams corresponds to the number of elements in a sequence.

The main idea of this paper is that n-grams can be obtained based on the order in which the elements appear in syntactic trees. Namely, we follow a path in the tree and construct n-grams, rather than taking them as they appear in the surface[2] representation of the text. Thus, we consider as neighbors the words (or other elements like POS tags, etc.) that follow one another in the path of the syntactic tree, and not in the text. We call such n-grams **syntactic n-grams (sn-grams)**. The great advantage of sn-grams is that they are based on syntactic relations of words and, thus, each word is bound to its "real" neighbors. This allows getting rid of the arbitrariness that is introduced by the surface structure. Note that syntactic n-grams are NOT n-grams constructed by using POS tags, as one can interpret in a naive fashion. In fact, this cannot be so strictly speaking, because POS tags represent morphological information, and not syntactic data. This paper is the extended version of the paper presented at MICAI 2012 (Sidorov, Velasquez, Stamatatos, Gelbukh, & Chanona-Hernandez, Syntactic Dependency-based N-grams as Classification Features, 2012).

So, the main idea of our proposal is related to the method of construction of these n-grams. In this manner, we get rid of surface language-specific information in sentences that causes problems while using traditional n-grams and maintain only persistent and pertinent linguistic information that has very clear interpretation. In our opinion, this is the way how syntactic information can be introduced into machine learning methods. Note that syntactic n-grams, though they are obtained in a different manner, keep being n-grams and can be applied practically in any task when traditional n-grams are used.

Obviously, there is a price to pay for using syntactic n-grams. Namely, parsing[3] should be performed for obtaining of the syntactic

* Corresponding author.
E-mail address: sidorov@cic.ipn.mx (G. Sidorov).

[1] Part of speech (POS) tags are labels which encode grammar classes and morphological information, for example, NN stands for "noun, singular", NNS for "noun, plural", etc.

[2] The term "surface" refers to the usual representation of texts, as contrasted to the "profound" representations, which are linguistic artifacts.

[3] Parsing is a process of the automatic construction of syntactic trees using specialized algorithms. Parser is software for parsing.

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.08.015

paths. There are various parsers available for many languages, but parsing takes time. Also, not for all languages there are parsers ready to use, though for well-studied languages like English or Spanish this consideration does not represent the problem. An interesting question for future work is evaluation of shallow parsing (i.e., incomplete parsing, not the whole tree is constructed, but only parts of it) usage. Shallow parsing is much faster than complete parsing, but is it enough for obtaining syntactic n-grams of good quality. We believe that for many tasks it can be sufficient. Another interesting question for future is if syntactic n-grams allow better comparison of results between languages. Obviously, during translation some syntactic relations are changed, but many of them are maintained. So, syntactic n-grams will be more "comparable" across the languages, since they try to get rid of the influence of language-specific surface structures.

In this paper, we apply sn-grams in authorship attribution problem using three popular classifiers and compare their performance using traditional n-grams as well. The experiments show better results for this task using sn-grams than using traditional n-grams. For this specific task, we also believe that sn-grams have real linguistic interpretation as far as the writing style of authors is concerned because they reflect real syntactic relations.

The rest of the paper is organized as follows: examples of syntactic n-grams are presented in Sections 2 and 3. The concept of sn-grams and their types are discussed in Section 4. The problem of authorship attribution is briefly introduced in Section 5. Then experimental results for authorship attribution based on syntactic n-grams are presented and compared with baseline sets of features in Section 6. Finally, we draw conclusions in Section 7.

2. Dependency and constituency trees in construction of syntactic N-grams

Let us consider a very simple example of application of sn-grams: a couple of similar phrases "*eat with wooden spoon*" vs. "*eat with metallic spoon*", see Fig. 1. Note that we can use both dependency and constituency representations of syntactic relations. They are equivalent if the head of each constituent is known. In our example of constituents, the heads are marked by heavier lines. It is very easy to add information about heads into constituency based grammar, namely, one of the components should be marked as the head in the corresponding rules.

In case of dependencies, we follow the path marked by arrows and obtain sn-grams. In case of constituents we first "promote" the head nodes so that they occupy the places in bifurcations, as

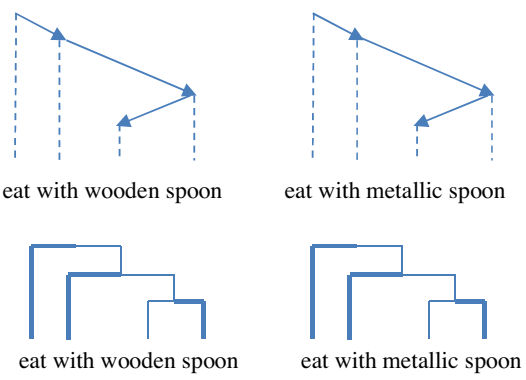

Fig. 1. Representations of syntactic relations.

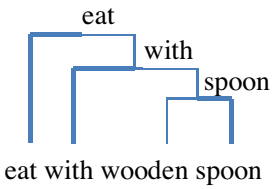

Fig. 2. Promoted head nodes.

shown in Fig. 2. Then we obtain sn-grams starting from the dependent constituencies and taking the heads from the bifurcations.

Let us consider the case of bigrams for this example. If we extract traditional bigrams from the phrases, then they have only one bigram in common: "*eat with*". Meanwhile, if we consider sn-grams, then two common bigrams are found: "*eat with*", "*with spoon*".

The same happens in case of trigrams. In case of traditional n-grams, these two phrases have no common n-grams, but if we use sn-grams then there is one common trigram: "*eat with spoon*".

In this example, sn-grams allow getting rid of the surface phenomenon of the English language that adds an adjective before the noun and in this way spoils traditional bigrams/trigrams. The same happens in case of, say, subordinate clauses, and, in general, in any type of syntactic structures.

Another possibility while obtaining sn-grams is to construct them ignoring auxiliary words (stop words). We follow the paths in the tree and just pass through the nodes of the stop words without adding them to the sn-gram. In case of our examples, the phrases have the common sn-bigram "*eat spoon*" (ignoring the preposition *the*) that cannot be obtained using traditional n-grams.

3. Full example of construction of Sn-grams

Now let us present a real-world example. We will discuss only dependency tree structure. We took the phrase from the novel "Dracula" by Bram Stocker: *I can even now remember the hour from which I dedicated myself to this great enterprise.* It has the following syntactic structure obtained by using the Stanford parser (de Marneffe, MacCartney, & Manning, 2006).

```
(ROOT
(S
  (NP (PRP I))
  (VP (MD can) („)
  (ADVP (RB even) (RB now))
    („)
    (VP (VB remember)
      (NP (DT the) (NN hour))
      (PP (IN from)
        (SBAR
        (WHNP (WDT which))
        (S
          (NP (PRP I))
          (VP (VBD dedicated)
          (NP (PRP myself))
          (PP (TO to)
          (NP (DT this) (JJ great) (NN
enterprise)))))))))
(. .)))
nsubj(remember-7, I-1)
aux(remember-7, can-2)
advmod(now-5, even-4)
advmod(remember-7, now-5)
```

root(ROOT-0, remember-7)
det(hour-9, the-8)
dobj(remember-7, hour-9)
prep(remember-7, from-10)
dobj(dedicated-13, which-11)
nsubj(dedicated-13, I-12)
pcomp(from-10, dedicated-13)
dobj(dedicated-13, myself-14)
prep(dedicated-13, to-15)
det(enterprise-18, this-16)
amod(enterprise-18, great-17)
pobj(to-15, enterprise-18)

Note that the number that corresponds to each node is not exactly its position; it is its position in the dependency tree, which can be different from the position in the sentence.

The Stanford parser generates two types of the output: tree structure, where the level of the component is represented by their indenting spaces, and syntactic relations that correspond to the dependency tree. For example, *dobj(remember-7, hour-9)* means that the word *remember* has 7th position in the tree and the word *hour* the 9th position, and that there is a relation *dobj* between them, where *remember* is the head word (it is the first one) and *hour* is the dependent word (the second one). We present the final syntactic tree in Figs. 3 and 4. In Fig. 3 we present only arrows that correspond to syntactic relations, while in Fig. 4 we also show labels of the syntactic relations. The Stanford parser handles 53 relations.

In this example, in our opinion, the parser committed an error, and connected the word *which* to *dedicated*, and not to *from*. We mark this by dashed arrow. Note that the small number of errors—if they are consistent—will not affect the performance of sn-grams (sure, it is better to have no errors). In further research we plan to analyze the impact of parser error in various tasks, i.e., the performance can also depend on the task in this case.

We hope that this example helps in understanding of the method of construction of sn-grams. Namely, we should traverse the tree node by node applying recursion in bifurcations. The method is intuitively very clear: follow the paths represented by arrows, take at each step words (or other elements) from the corresponding nodes and add them to the current n-grams that are under construction.

More formal description of the method is as follows. We start from the root node R. Then we choose the first arrow (we will pass through all arrows, so the order is not important) and take the node—let us denote it N—on the other side of the arrow. Our first bigram (or part of a larger sn-gram) is R–N. Note that the order is important because R is the head word and N is the dependent word. Now we move to the node N and repeat the operation, either

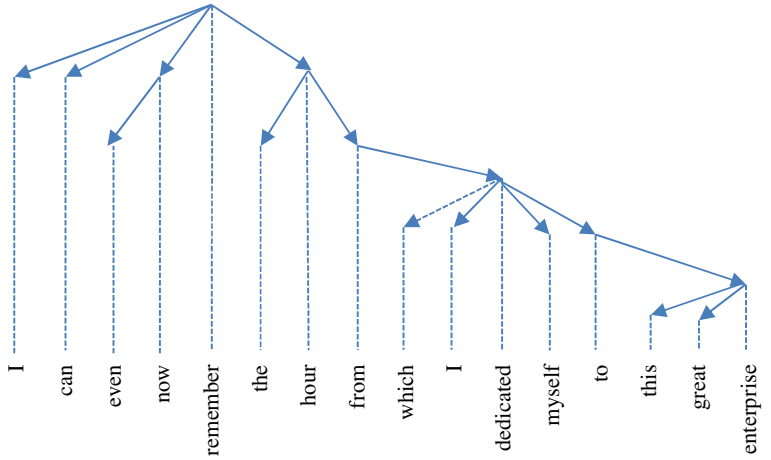

Fig. 3. Example of a syntactic tree.

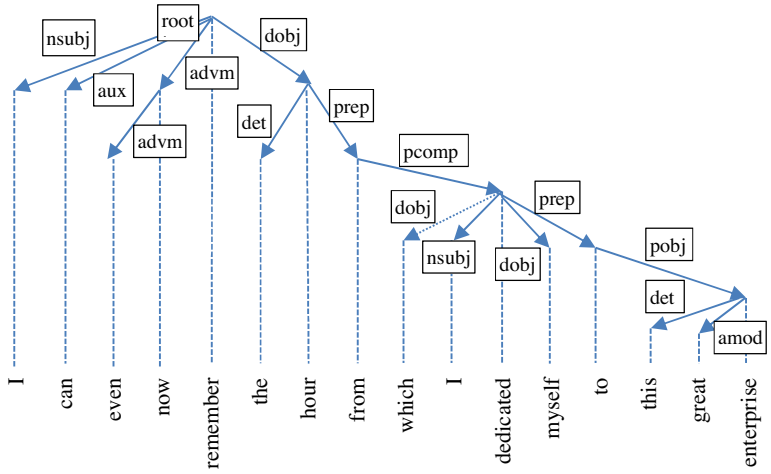

Fig. 4. Example of a syntactic tree with tags of syntactic relations.

Table 1
Syntactic and traditional bigrams from the example.

Syntactic bigrams	Traditional bigrams
Remember-now, now-even, remember-hour, remember-I, remember-can, hour-the, hour-from, dedicated-which, dedicated-I, from-dedicated, dedicated-myself, dedicated-to, to-enterprise, enterprise-this, enterprise-great	I-can, can-even, even-now, now-remember, remember-the, the-hour, hour-from, from-which, which-I, I-dedicated, dedicated-myself, myself-to, to-this, this-great, great-enterprise

Table 2
Syntactic and traditional bigrams from the example without stop words.

Syntactic bigrams	Traditional bigrams
Remember now, now-even, remember-hour, remember-dedicated, dedicated-enterprise, enterprise-great	Even-now, now-remember, remember-hour, hour-dedicated, dedicated-great, great-enterprise

Table 3
Syntactic and traditional trigrams from the example.

Syntactic bigrams	Traditional bigrams
Remember-now-even, remember-hour-the, remember-hour-from, hour-from-dedicated, from-dedicated-which, from-dedicated-I, from-dedicated-myself, from-dedicated-to, dedicated-to-enterprise, to-enterprise-this, to-enterprise-great	I-can-even, can-even-now, even-now-remember, now-remember-the, remember-the-hour, the-hour-from, hour-from-which, from-which-I, which-I-dedicated, I-dedicated-myself, dedicated-myself-to, myself-to-this, to-this-great, this-great-enterprise

Table 4
Syntactic and traditional trigrams without stop words from the example.

Syntactic bigrams	Traditional bigrams
Remember-now-even, remember-hour-dedicated, hour-dedicated-enterprise, dedicated-enterprise-great	Even-now-remember, now-remember-hour, remember-hour-dedicated, hour-dedicated-great, dedicated-great-enterprise

for the next bigram or for the larger sn-gram. The number of steps for construction of a sn-grams of a given size is equal to n-1, i.e., in case of bigrams we make only one step, in case of trigrams we make two steps, etc. In bifurcations, we apply recursion, i.e., each possible direction corresponds to a new sn-gram or a set of new sn-grams (in case that there are more bifurcations at lower levels). When we finish with the sn-gram, we return to the nearest previous bifurcation and continue in the direction that was not explored yet. Let us compare the results for extraction of traditional bigrams and syntactic bigrams for the example presented in Figs. 3 and 4. Note that the syntactic bigrams are presented directly in the output of the parser. (Table 1)

Now let us present syntactic and traditional bi-grams without stop words. Note that we continue the path if we meet the stop word, i.e., we do not stop, we just skip it and continue. (Table 2)

Here the especially interesting bigram is "*dedicated-enterprise*", where we skip several auxiliary words.

Finally, let us consider syntactic and traditional trigrams for the example (see Tables 3 and 4).

In our opinion, syntactic n-grams are much more stable and less arbitrary, i.e., they have more chances to be repeated in other sentences. We saw in Section 2 the simple example: what happens if we add an adjective for a noun. Traditional n-grams in the near context will be changed, but syntactic n-grams will maintain stable, only one new sn-gram will be added that corresponds to the pair *Noun–Adjective*.

Note that the number of syntactic bigrams is equal to the number of traditional bigrams. Let us see the reason. The number of traditional bigrams is equal to the number of words minus 1: for each word we consider its combination with the previous and the next word and get two bigrams, with exception for the first and the last words. For construction of sn-grams we consider syntactic arrows, being the number of these arrows exactly equal to the number of word minus 1, because each word has an incoming arrow—with exception of the root, and syntactic bigrams are formed by following each arrow. The number of sn-grams for $n > 2$ can be less or equal than in case of traditional n-grams. It is so because traditional n-grams consider just plain combinations, while for sn-grams there should exist "long" paths. It is particularly clear for greater values of n. Say, for $n = 7$, there are many n-grams in the above mentioned example, while there are only two sn-gram: *remember-hour-from-dedicated-to-enterprise-great/this.* It is obvious that the number of n-grams and sn-grams would be equal only if the whole phrase has exactly one path, i.e., there are no bifurcations.

Another possibility in construction on sn-grams is to use other elements than words, like POS tags or tags of syntactic relations (SR tags), see Section 4. Since we will use SR tags in experiments further in the paper, we would like to present, say, bigrams of SR tags extracted from the same example sentence*: root-nsubj, root-aux, root-advmod, admod-advmod, root-dobj, dobj-det, dobj-prep, prep-pcomp, pcomp-dobj, pcomp-dsubj, pcomp-prep, prep-pobj, pobj-det, pobj-amod.* In this case, we traverse the tree as well, but instead of nodes we take the names of arrows (arcs). In this work, we consider that SR tags are comparable with POS tags: they have similar nature and the quantity of both types of elements is relatively close—36 and 53 elements correspondingly.

4. Syntactic N-grams (sn-grams)

As we already explained, syntactic n-grams (sn-grams) are n-grams that are constructed using paths in syntactic trees. Their advantage is that they are much less arbitrary than traditional n-grams. Usually, for $n > 2$ their number is many less than the number of traditional n-grams, which simplifies their usage as classification features. Another advantage is that they can be interpreted as the linguistic phenomenon, while traditional n-grams have no plausible linguistic interpretation—they are merely statistical artifact.

The justification of the idea of sn-grams is related to introduction of linguistic information into statistically based methods of machine learning (classification). We believe that this idea helps to overcome the main disadvantage of traditional n-grams—they contain many arbitrary elements, i.e., a lot of noise, introduced by the surface structure.

The obvious disadvantage of syntactic n-grams is that previous syntactic processing is necessary. This consumes significant time and it is not easy to apply to some languages, because a syntactic parser and a set of lexical resources that are used by the parser are needed and not for any language these resources are available.

Previously, similar ideas were related to some specific tasks like using additional syntactic information in machine translation (Khalilov & Fonollosa, 2009) or generation in machine translation (Habash, 2004), without the generalization and taxonomy that we propose in this paper. The term syntactic n-gram is not very common and its importance is underestimated. It is used, for

example, in Agarwal, Biads, and Mckeown (2009) for extraction of polarity of syntactic constituents (chunks) as a whole element, but in the different sense.

Note that there are attempts to overcome the disadvantages of traditional n-grams using purely statistical approaches. We should mention skip-grams and Maximal Frequent Sequences (MFS).

Skip-grams (see, for example, Guthrie, Allison, Liu, Guthrie, & Wilks, 2006) are very similar to n-grams, but during their construction some elements of the corresponding sequence are ignored (skipped). It is an attempt to avoid possible noise, namely, by considering random variations in texts. There can be gaps (skips) with various skip steps.

An example of skip-grams: say, for the sequence ABCDE, we can obtain traditional bigrams AB, BC, CD, and DE. Skip-bigrams with the skip step of "one" will be AC, BD, and CE. Various skip steps can be used. Usually skip-grams also include traditional n-grams, in which case the skip step is zero. The problem with skip-grams is that their number grows very fast.

Maximal Frequent Sequences (MFS, see, for example, García-Hernández, Martínez Trinidad, & Carrasco-Ochoa, 2010) are skip-grams with major frequency, i.e., we take into account only skip-grams whose frequencies are greater than a certain threshold. The problem with MFS is that complex algorithms should be used for their construction and it takes substantial processing time. Another disadvantage of MSF is that unlike sn-grams, they depend on the text collection. And similar to skip-grams, no linguistic interpretation of MFS is possible in general case.

Now let us discuss what elements can form sn-grams. First of all it can be the elements that form traditional n-grams: words (stems, lemmas), POS tags, and characters. As we mentioned in Section 3, in case of syntactic n-grams, another type of elements can be used for their composition: tags of syntactic relations (SR tags), like *pobj, det, xcomp*, etc. These tags are similar to POS tags in the sense that they are morphosyntactic abstraction and are obtained during previous linguistic processing.

In case of the English language, we use the Stanford parser for determining SR tags, POS tags, and for construction of dependency-based syntactic trees. Although the parsing process was time consuming for our large corpus, it was performed only once, so the subsequent experiments do not take substantial time.

So, according to the **types of elements** that form syntactic n-grams, there can be various types of sn-grams:

- Word sn-grams: the elements of sn-gram are words (or stems, or lemmas, the important idea is that it is a lexical unit).
- POS sn-grams: the elements of sn-gram are POS tags.
- Sn-grams of tags of syntactic relations (SR tags): the elements of sn-gram are names of syntactic relations (see Section 3).
- Character sn-grams (see below).
- Mixed sn-grams: sn-grams are composed by mixed elements like words (lexical units), POS tags and/or SR tags. Since a sn-gram follows a syntactic link, there can be reasons for using mixed sn-grams, for example, they can reflect subcategorization frames. There are many possible combinations regarding to what part—the head word or the dependent word in the relation—should be represented by the lexical unit, the POS tag, or the SR tag. It seems that character sn-grams cannot be part of mixed sn-grams. These combinations should be explored experimentally in future.

In the paper Sidorov et al., Syntactic Dependency-based N-grams as Classification Features (2012), we mentioned that sn-grams of characters are impossible. Now we consider that they can be constructed from other syntactic n-grams. In this case, the main source would be sn-grams of words, though we can try stems or lemmas as well. So, we can construct sn-grams of characters from sn-grams of words. In general, the intuition behind the character n-grams is related with the two things: inside the words they should reflect some lexical phenomena; on the word boundaries they should reflect word combination patterns. It would be also interesting to apply these two possibilities separately. These ideas can be reflected in the character sn-grams in the same manner as in the traditional character n-grams. Still, it is an experimental work that should confirm if the character sn-grams are useful.

As far as the **treatment of stop words** is concerned, as we mentioned, they can be ignored or taken into account. This degree of freedom is always present in any preparation of linguistic data for further use with machine learning methods.

5. Task of the authorship attribution

The authorship attribution deals with an old and difficult question: how to assign a text of unknown or disputed authorship to a member of a set of candidate authors for whom undisputed samples of texts are available (Juola, Authorship Attribution, 2006). Despite its application to literary works, the rapid expansion of online text in Internet media (e.g., blogs, e-mail messages, social media postings, etc.), revealed practical applications of the authorship attribution are usually associated with forensic tasks (Abbasi & Chen, 2005).

The automated approaches to this problem involve the use of statistical or machine learning methods (Stamatatos, 2009). From the machine learning point of view, authorship attribution can be seen as a single-label multi-class classification task (Sebastiani, 2002). There are two basic steps: first, the texts should be appropriately represented as vectors of numerical values and, then, a classification algorithm can use these vectors to estimate the probability of class membership for any given text.

Stylometry is the line of research studying the quantification of writing style. So far, there are plenty of approaches that attempt to provide reliable solutions for capturing the properties of textual style (Holmes, 1998). The most successful and robust methods are based on low-level information such as character n-grams or auxiliary words (function word, stop words, e.g. articles, prepositions, etc.) frequencies (Stamatatos, 2009). Such measures are easy to get extracted from texts and can easily be applied to any given natural language. On the other hand, low-level features capture tiny pieces of stylistic information and require very high dimensionality of the representation that is really hard to be interpreted cumulatively. Therefore, they can hardly provide an in-depth understanding of authorial choices, so it is not easy for them to be used in tasks where the explanation of stylistic analysis is required (e.g., in cases of using authorship attribution results as evidence in court). In such cases, we need a more elaborate representation of style that is based on more abstract features coming from syntactic or semantic analysis of texts. In the framework of forensic applications where the aim is not only to find the most probable author of a text but also to explain this decision, the use of low-level features is problematic. However, the use of high-level information in authorship attribution experiments has not provided encouraging results since they seem to be less reliable in comparison to low-level features, although the combination of low-level and high-level features tends to improve the effectiveness (van Halteren, 2007).

Thousands of stylometric features have been proposed so far. These features can be distinguished in the following main categories according to the textual analysis they require (Stamatatos, 2009): lexical features (e.g., function word frequencies, word n-gram frequencies, vocabulary richness measures, etc.), character features (e.g., letter frequencies and character n-grams), syntactic features (e.g., part-of-speech tag frequencies, sentence and phrase

structure measures, rewrite rule frequencies etc.), semantic features (e.g., synonym measures, semantic dependency measures etc.), and application-specific features (e.g., font size, font color, specific word frequencies, etc.). Until now, a number of studies (Grieve, 2007; Keselj, Peng, Cercone, & Thomas, 2003; Luyckx, 2010) have shown that the most effective measures are lexical and character features. Although they are easy to be handled by a computational model, they provide little help on the explanation of the similarities/differences of writing style of different documents. Moreover, the effectiveness of all these measures depends on the text size. Modern text genres such as Twitter messages or SMS messages include very short texts (less than 200 words). In order to be able to handle short texts the distribution of each feature in the text (instead of a single measure) has been proposed (Escalante, Solorio, & Montes-y-Gomez, 2011), an approach that considerably increases the dimensionality of the representation.

The classification methods used in authorship attribution follow two main paradigms (Stamatatos, 2009). The profile-based methods treat each training text cumulatively (per author) (Keselj et al., 2003; Koppel, Schler, & Argamon, Authorship attribution in the wild, 2011). In more detail, they concatenate all the available training texts per author in one big file and extract a cumulative representation of that author's style (usually called the author's profile) from this concatenated text. That is, the differences between texts written by the same author are disregarded. On the other hand, the instance-based methods require multiple training text samples per author in order to develop an accurate attribution model (Diederich, Kindermann, Leopold, & Paass, 2003; Koppel, Schler, & Bonchek-Dokow, Measuring differentiability: unmasking pseudonymous authors, 2007). That is, each training text is individually represented as a separate instance of authorial style. Profile-based methods attempt to capture a general style for each author while instance-based methods capture a separate style of each individual document.

So far, two international competitions on authorship attribution have been organized (Argamon & Juola, 2011; Juola, Ad-hoc authorship attribution competition, 2004). In general, the results show that the effectiveness of state-of-the-art approaches can be very high given that the text size is not too short (at least 500 words) and the set of candidate author is relatively small (a dozen of authors). When very short texts are examined, the set of candidate authors grows large, or the open-set scenario (i.e., the true author is not necessarily included in the candidate author set) is adopted, the accuracy results significantly drop. However, the application of authorship attribution tools in difficult tasks indicates that they can perform better than an average human expert since they use more detailed representation of texts (e.g., character n-grams). On the other hand, this type of information is not easy to be understood by humans and this is a crucial obstacle in the exploitation of this technology in forensic applications.

Note that information about syntactic relations usually is not used in this task, being to some extent one of the exceptions (Baayen, Tweedie, & Halteren, 1996), where the authors analyze syntactic rewriting rules. In this paper, we show that syntactic relations represented as sn-grams are very effective feature set.

In this paper, we use as baseline features: character features, lexical features (words), and POS tags obtained using traditional n-grams, i.e., according to the appearance of the elements in texts.

6. Experimental results and discussion

The experiments were performed over corpus data for the authorship attribution problem. The corpus used in our study includes texts downloaded from the Project Gutenberg. We selected books of native English speaking authors that had their literary production in a similar period. In this paper, all experiments are conducted for the corpus of 39 documents by three authors. For evaluation of the experiments, we used 60% of the data for training, and the rest 40% for classification, as presented in Table 5. We use the term "profile size" for representing the first most frequent n-grams/sn-grams, e.g., for profile size of 400 only first 400 most frequent n-grams are used. We tested various thresholds for profile size and selected five thresholds for profile size as presented in all tables with the results.

The classification task consists in selecting features for constructing the vector space model, training supervised classification algorithms—we used WEKA for classification (Hall et al., 2009), and performing the classification—i.e., deciding to what class belongs the text fragment—in the given vector space model. In our previous work (Sidorov et al., Syntactic Dependency-based N-grams as Classification Features, 2012), we used only one classifier (classification algorithm)—SVM. In this work we present results for three classifiers: SVM (NormalizedPolyKernel of the SMO), Naive Bayes, and J48. They are described, for example, in WEKA manuals (Hall et al., 2009). Several baseline feature sets are analyzed that use traditional n-grams: words, POS tags and characters. Note that in general SVM is known to produce good results in the task of the authorship attribution. WEKA allows evaluation of the accuracy of the classification: correctly vs. incorrectly classified instances.

We used several features as baseline: word based features, character based features, and POS tags. For baseline features, we used traditional n-gram technique, i.e., the elements are taken as they appear in texts. We applied sn-grams technique for the same corpus with the results that outperform baseline methods.

When a table cell contains *NA (not available)*, it means that our data were insufficient to obtain the corresponding number of n-grams. It happens only with traditional bigrams of POS tags and characters, because the sets of characters and POS tags have relatively small size and there are less traditional bigrams than trigrams, etc. In these cases the total number of all bigrams is less than the given profile size.

In Tables 6–8 the results of the application of the baseline features using traditional n-grams are presented. Table 9 contains the results obtained using sn-grams.

If we compare the results of various classifiers, it can be observed that in the vast majority of cases the results of the SVM classification method are better than using NB and J48. We obtained the best performance with SVM always getting a 100% of accuracy with bigrams and trigrams for any profile size applying sn-grams. The only case when NB gives the same result as SVM (of 100%) is for the profile size of 400 for bigrams. Still, we consider that it is due to the fact that our topline can be achieved relatively easy because of the corpus size and the number of authors (only three). Note that in all other cases SVM got better results.

On the other hand, the tendency that sn-grams outperform traditional n-grams is preserved and allows us getting rid of the necessity to choose the threshold values, like the profile size for this corpus, for example, traditional POS trigrams got 100% for profile sizes of 4000 and 7000 only, while sn-grams (bigrams and trigrams) give 100% for all considered profile sizes.

Table 5
Training and classification data.

Author	Training		Classification	
	Novels	Size (MB)	Novels	Size (MB)
Booth Tarkington	8	3.6	5	1.8
George Vaizey	8	3.8	5	2.1
Louis Tracy	8	3.6	5	2.2
Total	24	11	15	6.1

Table 6
Word based n-grams, baseline.

Profile size	Classifier	n-Gram size			
		2 (%)	3 (%)	4 (%)	5 (%)
400	SVM	86	81	67	45
	NB	48	67	81	85
	J48	67	76	71	60
1000	SVM	86	71	71	48
	NB	76	81	95	90
	J48	71	67	71	67
4000	SVM	86	95	67	48
	NB	62	65	81	86
	J48	81	70	81	57
7000	SVM	86	90	71	45
	NB	52	48	81	81
	J48	86	71	86	51
11,000	SVM	89	90	75	33
	NB	53	52	90	78
	J48	89	81	70	44

Table 7
POS n-grams, baseline.

Profile size	Classifier	n-Gram size			
		2 (%)	3 (%)	4 (%)	5 (%)
400	SVM	90	90	76	62
	NB	67	62	57	52
	J48	76	57	52	71
1000	SVM	95	90	86	67
	NB	76	57	62	52
	J48	71	62	81	57
4000	SVM	NA	100	86	86
	NB	NA	57	62	57
	J48	NA	62	67	76
7000	SVM	NA	100	90	86
	NB	NA	38	62	57
	J48	NA	38	86	86
11,000	SVM	NA	95	90	86
	NB	NA	43	48	57
	J48	NA	57	86	90

Table 8
Character-based n-grams, baseline.

Profile size	Classifier	n-Gram size			
		2 (%)	3 (%)	4 (%)	5 (%)
400	SVM	90	76	81	81
	NB	71	62	71	67
	J48	76	62	48	76
1000	SVM	95	86	86	76
	NB	76	76	67	81
	J48	81	67	67	71
4000	SVM	90	95	90	86
	NB	90	71	81	71
	J48	76	76	90	81
7000	SVM	NA	90	86	86
	NB	NA	62	76	71
	J48	NA	76	86	90
11,000	SVM	NA	100	90	86
	NB	NA	67	62	71
	J48	NA	71	81	86

For better appreciation of the comparison of the results, we present Tables 10–13 , where the results are grouped by the size of n-grams/sn-grams applying SVM classifier. We show only SVM results since they are the best.

Table 9
Sn-grams of SR tags.

Profile size	Classifier	n-Gram size			
		2 (%)	3 (%)	4 (%)	5 (%)
400	SVM	100	100	87	93
	NB	100	93	73	67
	J48	87	67	93	73
1000	SVM	100	100	87	93
	NB	80	67	80	80
	J48	87	67	93	73
4000	SVM	100	100	93	73
	NB	40	40	53	60
	J48	67	47	73	73
7000	SVM	100	100	87	87
	NB	53	33	33	73
	J48	67	80	67	73
11,000	SVM	100	100	93	87
	NB	40	33	33	60
	J48	67	80	53	73

Table 10
Comparison for bigrams, SVM.

Profile size	Features			
	sn-Grams of SR tags (%)	n-Grams of POS tags (%)	Character based n-grams (%)	Word based n-grams (%)
400	100	90	90	86
1000	100	95	95	86
4000	100	NA	90	86
7000	100	NA	NA	86
11,000	100	NA	NA	89

Table 11
Comparison for trigrams, SVM.

Profile size	Features			
	sn-Grams ofSR tags (%)	n-Grams of POS tags (%)	Character based n-grams (%)	Word based n-grams (%)
400	100	90	76	81
1000	100	90	86	71
4000	100	100	95	95
7000	100	100	90	90
11,000	100	95	100	90

Table 12
Comparison for 4-grams, SVM.

Profile size	Features			
	sn-Grams ofSR tags (%)	n-Grams of POS tags (%)	Character based n-grams (%)	Word based n-grams (%)
400	87	76	81	67
1000	87	86	86	71
4000	93	86	90	67
7000	87	90	86	71
11,000	93	90	90	75

It can be appreciated that in all cases sn-gram technique outperforms the technique based on traditional n-grams. We consider that SR tags and POS tags are similar for the purposes of our comparison; both are small sets of tags: 36 and 53 elements associated with words.

Table 13
Comparison for 5-grams, SVM.

Profile size	Features			
	sn-Grams ofSR tags (%)	n-Grams of POS tags (%)	Character based n-grams (%)	Word based n-grams (%)
400	93	62	81	45
1000	93	67	76	48
4000	73	86	86	48
7000	87	86	86	45
11,000	87	86	86	33

As can be seen, topline of our task is very high: 100%. It is related to the fact that we use much data and our classification only distinguishes between three classes. In some cases, baseline methods also reach the topline. Still, it happens only for a small number of specific profile sizes. The best results are obtained by sn-grams using bigrams and trigrams for any profile size. For any combination of parameters baseline methods got worse results than sn-grams.

The question can arise if it is worth working with the small number of classes. In our opinion, it is useful and important. First of all, the authorship attribution is often a real world application in case of a dispute over the authorship of a document, and in this case the number of classes is reduced to two or three, i.e., it is our situation. We started experiments with more classes (seven) as described in Sidorov, Velasquez, Stamatatos, Gelbukh, & Chanona-Hernandez, Syntactic Dependency-Based N-grams: More Evidence of Usefulness in Classification (2013).

7. Conclusions and future work

In this paper, we introduce a concept of syntactic n-grams (sn-grams). The difference between traditional n-grams and sn-grams is related to the manner of which elements are considered neighbors. In case of sn-grams, the neighbors are found by following syntactic relations in syntactic trees, while traditional n-grams are formed as the words appear in texts. The concept of sn-grams allows bringing syntactic information into machine learning methods. Sn-grams can be applied in all tasks when traditional n-grams are used.

Any syntactic representation can be used for application of sn-gram technique: dependency trees or constituency trees. In case of dependency tree, we should follow the syntactic links and obtain sn-grams. In case of constituency trees, some additional steps should be made, but these steps are very simple.

We conducted experiments for authorship attribution task using SVM, NB, and J48 for several profile sizes. Relatively large corpus of works of three authors was used. We used as the baseline features traditional n-grams of words, POS tags and characters. The results show that sn-gram technique outperforms the baseline technique.

The following directions of future work can be mentioned:

- Experiments with all feature sets on larger corpus (and more authors, i.e., more classes).
- Analysis of the applicability of shallow parsing instead of full parsing.
- Analysis of usefulness of sn-grams of characters.
- Analysis of the impact of parser errors on the performance of sn-grams.
- Analysis of behavior of sn-grams between languages, e.g., in parallel texts or comparable texts.
- Application of sn-grams in other NLP tasks.
- Application of mixed sn-grams.
- Experiments that would consider combinations of the mentioned features in one feature vector.
- Evaluation of the optimal number and size of sn-grams for various tasks.
- Consideration of various profile sizes with more granularity.
- Application of sn-grams in other languages.

Acknowledgments

Work done under partial support of Mexican government (projects CONACYT 50206-H and 83270, SNI) and Instituto Politécnico Nacional, Mexico (projects SIP 20111146, 20113295, 20120418, COFAA, PIFI), Mexico City government (ICYT-DF project PICCO10-120) and FP7-PEOPLE-2010-IRSES: Web Information Quality - Evaluation Initiative (WIQ-EI) European Commission project 269180. We also thank Sabino Miranda and Francisco Viveros for their valuable and motivational comments.

References

Abbasi, A., & Chen, H. (2005). Applying authorship analysis to extremist-group web forum messages. *IEEE Intelligent Systems, 20*(5), 67–75.

Agarwal, A., Biads, F., & Mckeown, K. (2009). Contextual phrase-level polarity analysis using lexical affect scoring and syntactic N-grams. In *Proceedings of 12th conference of the European chapter of the ACL (EACL)* (pp. 24–32).

Argamon, S., & Juola, P. (2011). Overview of the international authorship identification competition at PAN-2011. In *Proceedings of fifth international workshop on uncovering plagiarism, authorship, and social software misuse*.

Baayen, A., Tweedie, F., & Halteren, H. (1996). Outside the cave of shadows: Using syntactic annotation to enhance authorship attribution. *Literary and Linguistic Computing*, 121–131.

de Marneffe, M., MacCartney, B., & Manning, C. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Diederich, J., Kindermann, J., Leopold, E., & Paass, G. (2003). Authorship attribution with support vector machines. *Applied Intelligence, 19*(1), 109–123.

Escalante, H., Solorio, T., & Montes-y-Gomez, M. (2011). Local histograms of character n-grams for authorship attribution. In *Proceedings of 49th annual meeting of the association for computational linguistics* (pp. 288–298).

García-Hernández, R., Martínez Trinidad, J., & Carrasco-Ochoa, J. (2010). Finding maximal sequential patterns in text document collections and single documents. *Informatica, 34*(1), 93–101.

Grieve, J. (2007). Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing, 22*(3), 251–270.

Guthrie, D., Allison, B., Liu, W., Guthrie, L., & Wilks, Y. (2006). A closer look at skip-gram modelling. In *Proceedings of LREC*.

Habash, N. (2004). The use of a structural N-gram language model in generation-heavy hybrid machine translation. *LNCS, 3123*, 61–69.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. (2009). The WEKA data mining software: An update. *SIGKDD Explorations, 11*(1).

Holmes, D. (1998). The evolution of stylometry in humanities scholarship. *Literary and Linguistic Computing, 13*(3), 111–117.

Juola, P. (2004). Ad-hoc authorship attribution competition. In *Proceedings of the joint conference of the association for computers and the humanities and the association for literary and linguistic computing* (pp. 175–176).

Juola, P. (2006). Authorship attribution. *Foundations and Trends in Information Retrieval, 1*(3), 233–334.

Keselj, V., Peng, F., Cercone, N., & Thomas, C. (2003). N-gram-based author profiles for authorship attribution. *Computational Linguistics*, 225–264.

Khalilov, M., & Fonollosa, J. (2009). N-gram-based statistical machine translation versus syntax augmented machine translation: Comparison and system combination. In *Proceedings of 12th conference of the European chapter of the ACL* (pp. 424–432).

Koppel, M., Schler, J., & Argamon, S. (2011). Authorship attribution in the wild. *Language Resources and Evaluation, 45*(1), 83–94.

Koppel, M., Schler, J., & Bonchek-Dokow, E. (2007). Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 1261–1276.

Luyckx, K. (2010). *Scalability Issues in Authorship Attribution.* Ph.D. thesis, University of Antwerp.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys, 34*(1).

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernandez, L. (2012). Syntactic dependency-based N-grams as classification features. In *Proceedings of MICAI: Vol. 7630. LNAI* (pp. 1–11).

Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., & Chanona-Hernandez, L. (2013). Syntactic dependency-based N-grams: More evidence of usefulness in classification. In *Proceedings of CICLing: Vol. 7816. LNCS* (pp. 13–24).

Stamatatos, E. (2009). A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology, 60*(3), 538–556.

van Halteren, H. (2007). Author verification by linguistic profiling: An exploration of the parameter space. *ACM Transactions on Speech and Language Processing, 4*(1), 1–17.