Applying Kernel Methods to Argumentation Mining

Niall Rooney, Hui Wang
AI & Applications Research Group
School of Computing & Mathematics
University of Ulster
email:{nf.rooney, h.wang}@ulster.ac.uk

Fiona Browne
Northern Ireland Technology Center
Queen's University Belfast
email:f.browne@qub.ac.uk

Abstract

The area of argumentation theory is an increasingly important area of artificial intelligence and mechanisms that are able to automatically detect the argument structure provide a novel area of research. This paper considers the use of kernel methods for argumentation detection and classification. It shows that a classification accuracy of 65%, can be attained using Natural Language Processing based kernel approaches, which do not require any heuristic choice of features.

Introduction

Argumentation involves the detection of various elements of an argument as outlined in Toulmin's influential model of argumentation (Toulmin 1958). An argument is a set of premises, pieces of evidence (e.g. facts), offered in support of a claim. The claim is a proposition, an idea which is either true or false, put forward by somebody as true. The claim of an argument is also normally called its conclusion. Argumentation may also involve chains of reasoning, where claims are used as premises for deriving further claims. In the argumentation process, relationships between these elements are built by leveraging Natural Language Processing and Computational Reasoning. A number of important approaches have been applied within the field to formalize the process of argumentation. In Classical Logic approaches to argumentation, arguments are represented as a set of facts and conclusion(s) that follows from the given facts (Bench-Capon and Dunne 2007). As there is no variation in the weight an individual fact can have, uncertainty can be represented as disjunctions. Classical Logic is a well-established formalism which allows the use of tools such as highly mature and efficient theorem provers. Once a knowledge base has been created, a range of operations can be performed. For each argument, counterarguments can be found.

Similar to (Palau and Moens 2009; 2011), we consider an argument as a set of elementary units or propositions, being all of them premises, except the ultimate one,which is the conclusion and our focus is being able to automatically detect the premises and conclusion from all sentences of the free text containing the argument. As a sentence (particularly in a formal argument) is normally composed of main and/or subordinate clauses, with the argumentation element contained within one or more of the clauses, our classification approach focuses on whether a sentence contains in part or whole an argumentative element or not. Rather than consider a range of possible NLP (Natural Language Processing) based features often chosen based on heuristic considerations, we consider the use of convolution kernels for this task, which have been shown to yield good performance for a range of NLP tasks (Zhang, Zhang, and Li 2010). A kernel provides a mechanism for calculating the similarity between two objects e.g. a document, a paragraph, sentence etc. Formally a kernel $k(x, y)$ is a similarity measure defined by an implicit mapping Φ, from the original space to a vector space (feature space) such that: $k(x, y) = \Phi(x) \cdot \Phi(y)$. A convolution kernel is a recursively constructed kernel for a structured object such that the kernel is the composition of kernels applied to parts of the object (Collins and Duffy 2001).

Kernel Methods

There has been a number of directions of research into the use of convolution kernel methods for NLP based on the sequence output from tagging or constituency and/or dependency tree output parsing. In this paper we consider only the former but the reader should refer to extensive work by Moschitti for the latter over a range of tasks such as relation extraction (Nguyen, Moschitti, and Riccardi 2009) to cite but one. One approach for representing the similarity of one sentence or sub-sentence (or clause) to another is to compare the sequence of tagged features for each word to another sequence, where possible tags (or symbols) may refer to the word form itself, its root and its Part of Speech (POS). Sequence kernels ((Lodhi et al. 2002; Cancedda et al. 2003)) assess similarity by counting the number of (possibly non-contiguous) matching subsequences of tags. Let x be a sequence of $|x|$ symbols drawn from an alphabet Σ e.g. in the case of word forms, Σ denotes the set of possible word forms. A subsequence s_x of length n is identified by a set of n symbol indices constrained to be strictly increasing $s_x = x[i]$ with $i \in \{1, .., |x|\}$ and $i[1] < i[2] \cdots < i[n]$. Let $\mathcal{S}_n(x)$ be the set of subsequences of size n associated to x and we denote as $g(s_x)$ as the number of gaps contained in the subsequence s_x. The gap-weighted subsequence kernel of order n is defined as :

$$k_n(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \mathbf{1}(s_x = s_y) \lambda^{g(s_x)} \lambda^{g(s_y)} \quad (1)$$

where λ is referred to as the gap penalization factor. When λ tends to zero, the influence of non-contiguous subsequences is diminished. For $\lambda = 1$, every subsequence contributes equally to the kernel regardless of whether it is contiguous or not.

For all subsequences up to and including size n, the"blended" kernel K is defined as:

$$K = \sum_{i=1}^{n} \mu_i k_i(x,y) \quad (2)$$

The value of μ_i is set to one in the context of this paper so that all kernel contributions for different orders of size are equally weighted, as there is no particular preference for matching subsequences of different sizes.

Cancedda and Mahé (2009) extended the concept of sequences to include p multiple dimensions or factors. Under this representation, a symbol u is formally defined as a tuple of p factors $\{u^{(d)}\}_{d=1:p}$ each drawn from a different alphabet Σ_d .

It can be shown that:

$$k_n^{fact}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{g(s_x)} \lambda^{g(s_y)} \prod_{i=1}^{n} \sum_{d=1}^{p} w_d \mathbf{1}(s_x^{(d)}[i] = s_y^{(d)}[i]) \quad (3)$$

where d denotes each dimension and w_d is the weighting for each dimension. $s_x^{(d)}[i]$ denotes the factor value at position i in the subsequence for dimension d. For the purposes of this work, the factors are based on the symbols or tags representing the word form, its root or lemma and POS in that order, so that the value of p is 3.

For efficiency of computation, rather than calculate the gaps in a sequence, we determines the length of the subsequence in the original sequence according to the proposed mechanism of Bunescu and Mooney (2005) so that

$$k_n^{fact}(x,y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \lambda^{l(s_x)} \lambda^{l(s_y)} \prod_{i=1}^{n} \sum_{d=1}^{p} w_d \mathbf{1}(s_x^{(d)}[i] = s_y^{(d)}[i]) \quad (4)$$

where $l(s_x)$ is the length of the subsequence indices in the original sequence x i.e. $l(s_x) = i[n] - i[1] + 1$. This means that there is a penalization even for subsequences of length 1, however it is still diminished in comparison to longer (sparser) sub-sequences.

For all subsequences up to size n, the factored sequence kernel K^{Fact} is defined as:

$$K^{Fact} = \sum_{i=1}^{n} \mu_i k_i^{fact}(x,y) \quad (5)$$

Experimental Evaluation

In this section, we describe the corpus, the evaluation of argumentation mining carried out and the results/analysis of our experiments.

Corpus

The AraucariaDB (Reed et al. 2008) was developed as part of a project at the University of Dundee (UK) and comprises a range of argumentative examples drawn from a number of different regions and sources. Each document has been annotated; marked up in an XML style format using Araucaria , a software tool according to known argumentation schemes (Walton, Reed, and Macagno 2008). In total there is at the time of this study, 662 documents. An example of the argument elements drawn from a sample argument document is shown in Figure 1.

```
<ARG>
  <AU>
    <PROP identifier='A' missing='no'>
      <PROPTEXT offset='10'>it matters if WMD are found</PROPTEXT>
      <INSCHEME scheme='Argument from the Constitution of
             Properties' schid='0' />
    </PROP>
    <LA>
      <AU>
        <PROP identifier='B' missing='no'>
          <PROPTEXT offset='39'>The reason given for invading Iraq
                now was because they were an imminent threat
                because of their weapons of mass destruction
        </PROPTEXT>
          <INSCHEME scheme='Argument from the Constitution of
                Properties' schid='0' />
        </PROP>
      </AU>
      <AU>
        <PROP identifier='C' missing='yes'>
          <PROPTEXT offset='1'>If the reason given for invading
                Iraq was that its WMD were an imminent threat, it
                matters if WMD are found
          </PROPTEXT>
          <INSCHEME scheme='Argument from the Constitution of
                Properties' schid='0' />
        </PROP>
      </AU>
    </LA>
  </AU>
  <EDATA>
    <AUTHOR>Mark</AUTHOR>
    <DATE>2003 09 05</DATE>
    <SOURCE>BBC News, Have your say, 'WMD: Should there be an
          independent inquiry?' 10 July 2003.</SOURCE>
    <COMMENTS />
  </EDATA>
</ARG>
```

Figure 1: Sample Araucaria argument listing

AU denotes an argument element where the first argument refers to the Conclusion and all subsequent argument elements if the attribute ”missing” has value no, refers to known premises. So in the given example , there is only one premise: “The reason given for invading Iraq now was because they were an imminent threat because of their weapons of mass destruction”. We processed this corpus using GATE (Cunningham et al. 2002) to annotate premises and conclusions in the corpus and to determine each sentence and its associated label. A sentence in the text of any document is labelled as a “Premise” if it contains a part or all of a premise only, a “Conclusion” if it contains part or all of a Conclusion only, or “None” indicating that the sentence does not belong to any argumentative element. The evaluation label “PremiseConclusion” denotes a sentence that is both part of a Premise and part of a Conclusion, as Premise and Conclusion boundaries do not necessarily coincide with Sentence boundaries. Also the classification of “Conclusion” is more difficult than a “Premise” as the level of distinct-ness amongst conclusions is higher as an argument has at most only one argument whereas it may have many premises. Table 1 shows the number of sentence instances containing each label.

Label	Number of instances
None	1686
Conclusion	304
Premise	1299
PremiseConclusion	161

Table 1: Number of instances of each argumentative element label

Evaluation

We developed a GATE plugin in order to run 10 fold cross validation to apply a SVM classifier using a kernel function based on the normalized factored sequence kernel in order to classify the label of a sentence, based on the previously mentioned labels. In the case of the factored sequence kernel, we considered different settings for the maximum length of subsequence, the gap weighting penalization factor and the relative weighting setting for each factor, by varying each parameter while keeping the other parameters constant, to determine the effect on classification performance.

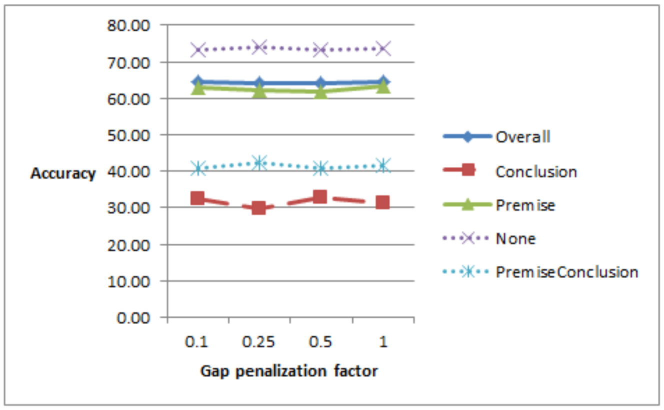

Figure 2: The effect on cross validation accuracy by varying the gap penalization size

Figure 2 shows the effect on the cross validation accuracy by varying λ where w_d for each factor is 1 and size n is 2. Noticeably the accuracy for “Conclusion” is low , a reflection of the fact that as Table 1 shows there is large imbalance in the number of sentences that are “Conclusion” to sentences containing label “None” or “Premise”. A similar effect is shown with the sentences labelled as “PremiseConclusion”. In terms of the overall accuracy (as an average of various labellings) , there did not seem to be any benefit of having a gap weighting penalization factor less than 1, however it did not have a large detriment on the performance.

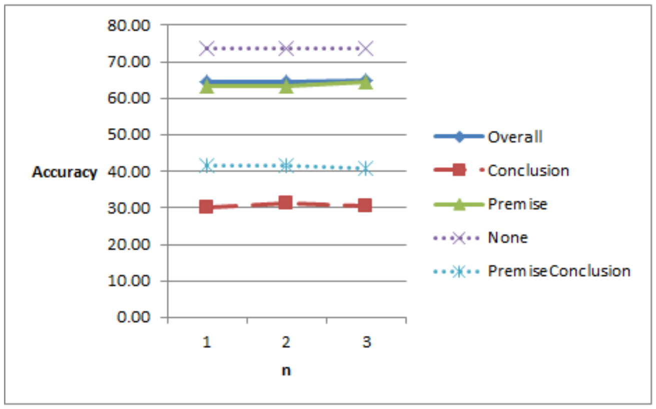

Figure 3: The effect on cross validation accuracy by varying the maximum size of n

Based on a gap penalization factor of 1, we investigated the effect of different sizes of n from 1 to 3. Figure 3 shows the outcome of this evaluation. It is clear that increasing the effect of n has only a slight improvement in accuracy, indicating that sentences in different arguments, do not have relatively high level of commonality in their higher order subsequences, in comparison to subsequences of only 1 element.

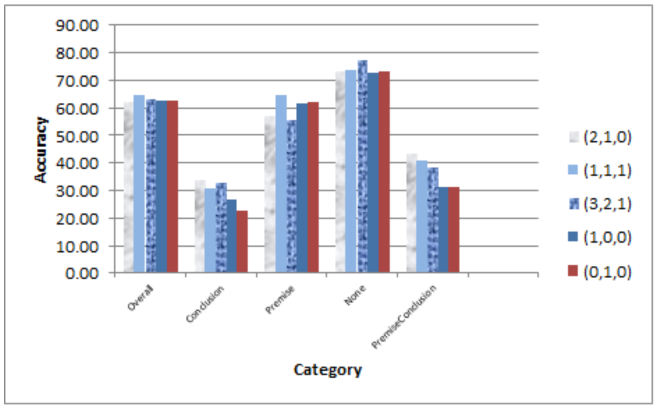

Figure 4: The effect on cross validation accuracy for different weighting factors (w_1 (word),w_2 (root), w_3 (pos))

Potentially this may be due to the high level of diversity of content due to the varying sources for arguments, and that higher order sub-sequences may only make a difference for content drawn only from specific sources such as court reports. This also explains to some extent the lack of a pronounced gap weighting effect. It may be neccessary to consider the classification of smaller units of text such as the use of clauses (Palau and Moens 2009).

Furthermore, based on a gap penalization factor of 1, we investigated the effect of different weightings for each of the given factors. Figure 4 shows the outcomes for three different weightings. An interesting effect that can be observed from the given figure is that the use of the POS is relatively inconsequential in its effect on performance, as shown by the weighting of zero. Including the factor results in only a slight increase in performance. Increasing the relative weighting of the word to the root, appears benefical for certain categories but detrimental for others.

Conclusions

We have investigated the use of convolution kernel methods for classifiying whether a sentence belongs to an argumentative element or not. Using gap-weighted subsequence kernels, we achieved an accuracy of 65%. This is less than the value of 73% stated in (Palau and Moens 2009). However we had the advantage that we did require the specialized selection of different lexical or syntactic based features, the selection of which are based on *ad hoc*s choices. Where argumentative elements are drawn from the same domain it is probable that kernel approaches will prove more benefical as more shared structures are involved either in terms of higher order sub-sequences in the case of sequence kernels, which we intend to look at more closely in future work. Another avenue of research is to consider classifying smaller argumentative units of text such as clauses.

References

Bench-Capon, T. J. M., and Dunne, P. 2007. Argumentation in artificial intelligence. *Artificial Intelligence* 171:619–641.

Bunescu, R., and Mooney, R. J. 2005. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. The Association for Computational Linguistics.

Cancedda, N., and Mahé, P. 2009. Factored sequence kernels. *Neurocomputing* 72(7-9):1407–1413.

Cancedda, N.; Gaussier, E.; Goutte, C.; and Renders, J. 2003. Word-sequence kernels. *Journal of Machine Learning Research* 3(6):1059–1082.

Collins, M., and Duffy, N. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, 625–632.

Cunningham, H.; Maynard, D.; Bontcheva, K.; and Tablan, V. 2002. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 168–175. Stroudsburg, PA, USA: Association for Computational Linguistics.

Lodhi, H.; Saunders, C.; Shawe-Taylor, J.; Cristianini, N.; and Watkins, C. 2002. Text classification using string kernels. *Journal of Machine Learning Research* 2:419–444.

Nguyen, T.; Moschitti, A.; and Riccardi, G. 2009. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, 1378–1387. Association for Computational Linguistics.

Palau, R. M., and Moens, M. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, 98–107.

Palau, R., and Moens, M.-F. 2011. Argumentation mining. *Artif. Intell. Law* 19(1):1–22.

Reed, C.; Mochales Palau, R.; Rowe, G.; and Moens, M. 2008. Language resources for studying argument. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*.

Toulmin, S. E. 1958. *The Uses of Argument*. Cambridge University Press.

Walton, D.; Reed, C.; and Macagno, F. 2008. *Argumentation Schemes*. Cambridge University Press.

Zhang, M.; Zhang, H.; and Li, H. 2010. Convolution kernel over packed parse forest. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 875–885. The Association for Computational Linguistics.

Acknowledgments

The DEEPFLOW project is funded by Invest Northern Ireland ref: RD1208002 and SAP (AG). We acknowledege the contribution of Fergal Monaghan and Zhiwei Lin of SAP who provided the foundation for this work.