Learning to Recognize Ancillary Information for Automatic Paraphrase Identification

Simone Filice
DICII,
University of Roma, Tor Vergata
filice@info.uniroma2.it

Alessandro Moschitti[*]
ALT, Qatar Computing Research Institute,
HBKU
amoschitti@qf.org.qa

Abstract

Previous work on Automatic Paraphrase Identification (PI) is mainly based on modeling text similarity between two sentences. In contrast, we study methods for automatically detecting whether a text fragment only appearing in a sentence of the evaluated sentence pair is important or ancillary information with respect to the paraphrase identification task. Engineering features for this new task is rather difficult, thus, we approach the problem by representing text with syntactic structures and applying tree kernels on them. The results show that the accuracy of our automatic Ancillary Text Classifier (ATC) is promising, i.e., 68.6%, and its output can be used to improve the state of the art in PI.

1 Introduction

Automatic PI is the task of detecting if two texts convey the same meaning. For example, the following two sentences from the Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2004):

S$_{1a}$: *Although it's unclear whether Sobig was to blame, The New York Times also asked employees at its headquarters yesterday to shut down their computers because of "system difficulties."*

S$_{1b}$: *The New York Times asked employees at its headquarters to shut down their computers yesterday because of "computing system difficulties."*

are paraphrases, while these other two are not:

S$_{2a}$: *Dr. Anthony Fauci, director of the National Institute of Allergy and Infectious Diseases, agreed.*

S$_{2b}$: *"We have been somewhat lucky," said Dr. Anthony Fauci, director of the National Institute of Allergy and Infectious Diseases.*

Most previous work on automatic PI, e.g., (Madnani et al., 2012; Socher et al., 2011), is based on a direct comparison between the two texts, exploiting different similarity scores into a machine learning framework. However, these methods consider sentences as monolithic units and can thus be misled by ancillary information that does not modify the main meaning expressed in the text.

For example, the additional text fragment (ATF), "*Although it's unclear whether Sobig was to blame*", from S_{1a} expresses ancillary information, which does not add much to the message of S_{1b}, thus the sentences are considered paraphrases. In contrast, S$_{2b}$ contains the ATF, "*We have been somewhat lucky*", whose meaning is not linked to any constituent of S$_{1b}$. Since such text expresses relevant information, the two sentences are not considered paraphrases.

In this paper, we study and design models for extracting ATFs from a sentence with respect to another one and classifying if their meaning is ancillary or important. For this purpose, we built a corpus of sentence pairs using MSRP, where at least one pair member always contains ATFs. We use SVMs with tree kernels applied to syntactic representations (Severyn and Moschitti, 2012) of ATFs for learning automatic ATCs.

The results derived on MSRP show (*i*) a promising accuracy of our ATC and (*ii*) the output of ATC

[*] Professor at DISI, University of Trento.

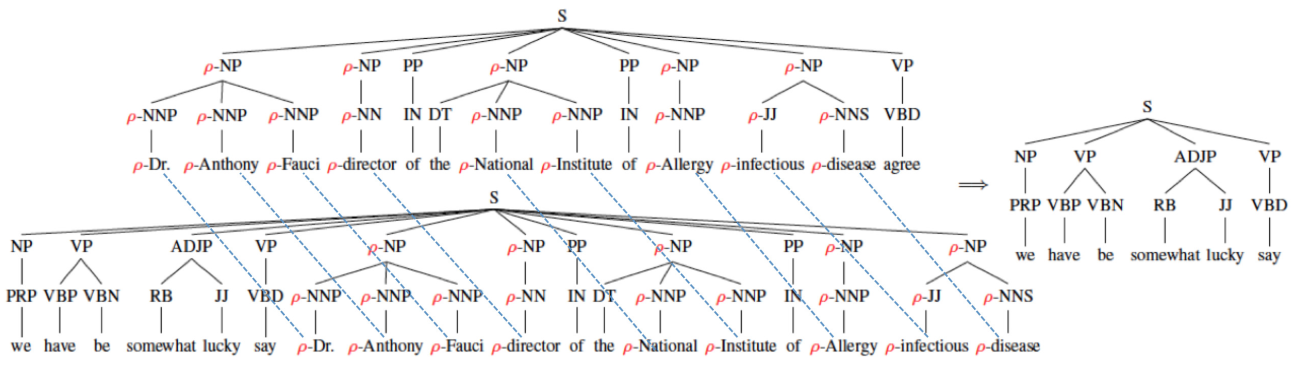
S
ρ-NP
ρ-NNP ρ-NNP ρ-NNP
ρ-Dr. ρ-Anthony ρ-Fauci
ρ-NP PP ρ-NP PP ρ-NP ρ-NP VP
ρ-NN IN DT ρ-NNP ρ-NNP IN ρ-NNP ρ-JJ ρ-NNS VBD
ρ-director of the ρ-National ρ-Institute of ρ-Allergy ρ-infectious ρ-disease agree
S
NP VP ADJP VP ρ-NP ρ-NP PP ρ-NP PP ρ-NP ρ-NP
PRP VBP VBN RB JJ VBD ρ-NNP ρ-NNP ρ-NNP ρ-NN IN DT ρ-NNP ρ-NNP IN ρ-NNP ρ-JJ ρ-NNS
we have be somewhat lucky say ρ-Dr. ρ-Anthony ρ-Fauci ρ-director of the ρ-National ρ-Institute of ρ-Allergy ρ-infectious ρ-disease
⟹
S
NP VP ADJP VP
PRP VBP VBN RB JJ VBD
we have be somewhat lucky say

Figure 1: A pair of non-paraphrase sentences and its corresponding additional fragment.

can be used as a feature for improving the state-of-the-art PI model.

2 Ancillary clauses in PI

Our main purpose in studying computational approaches to the detection of ancillary information is its practical application to PI. Thus, given a pair of sentences (in general two texts), we define ATFs as ancillary information if their semantics:

(*i*) only appears in one of the two sentences and

(*ii*) does not change the main meaning of the sentence, i.e., either the sentences are paraphrases or, if they are not, such ATFs are not the reason for their different meaning.

The definition above along with a syntactic representation of the sentences can be applied to a paraphrase corpus to build a dataset of ancillary vs. important ATFs. For example, Fig. 1 shows the shallow tree representation[1] we proposed in (Severyn and Moschitti, 2012) of the sentences, S$_{2a}$ and S$_{2b}$, reported in the introduction, where the red label ρ indicates that there is a link between the lemmas of the two sentences (also shown by the dashed edges). ρ can be propagated to the upper nodes to mark the related constituents. For example, the lemma *National* is matched by the two sentences, thus both its father node, NNP, and its grandfather constituent, NP, are marked.

Such representation makes the extraction of ATFs easier. For instance, Fig. 1 shows the text fragment of S_{2b}, "*We have been somewhat lucky*", on the right. This is an ATF since it is not aligned with any fragments of S_{2a}. Moreover, since it expresses a central information to the sentence meaning, S_{2b} cannot be in paraphrase relation with S_{2a}. Conversely, the ATF of S_{1a}, "*Although it's unclear whether Sobig was to blame*", is ancillary to the main meaning of the sentence, indeed, the annotators marked S_{1a} and S_{1b} as a valid paraphrase.

3 Building the ATC corpus

The previous section has shown an approach to extract ATFs that can be potentially ancillary. This uses an alignment approach based on lexical similarity, which may fail to align some text constituents. However, these mistakes only affect the precision in extracting ATFs rather than the recall. In other words, we can build a corpus that considers most cases of additional information.

In particular, we design the following simple heuristic: let F_i and F_j be the largest not aligned (possibly discontinuous) word sequences appearing in the sentence pair (S_i, S_j), where $F_i \in S_i$ and $F_j \in S_j$. We define ATF as the largest text between F_i and F_j subject to $d = |size(F_i) - size(F_j)| > \tau$, where $size(F)$ is the number of words[2] appearing in F. If the condition is not satisfied no ATF is extracted.

The condition over d is important because the sentence aligner may fail to match some subsequences, creating false ATFs. However, what is missed from one sentence will be missed also in the other sen-

[1]The shallow trees are constituted by four levels: (*i*) word lemmas as leaves, (*ii*) POS-tags as parent of lemmas, (*iii*) phrases grouping POS-tag nodes and (*iv*) a final root S.

[2]Only verbs, nouns, adjectives, adverbs and numbers were considered, assuming all the others as "not informative words".

	Train			Test		
τ	Ancillary	Important	Total	Ancillary	Important	Total
1	971	687	1658	387	687	1074
2	426	364	790	166	151	317
3	166	169	335	62	79	141
4	59	73	132	21	36	57

Table 1: Number of additional clauses extracted from MSRP.

tence. Thus, in general, if we set a small d then F_i and F_j misalignments may generate false ATFs. In contrast, a large d would clearly prevent this problem, although small ATFs (of size $< d$) may be discarded. More precisely, smaller values of τ may cause the selection of fragments that have corresponding fragments in the other sentence, expressed with dissimilar words (i.e., the aligner failed to match those constituents). Larger values of τ make the heuristic more precise, but less effective in retrieving smaller ATFs.

The ATF corpus. We applied the heuristic above to extract an ATF (if exists) from each sentence pair of MSRP. The number of the extracted ATFs depends on τ as reported in Table 1. A manual inspection of the retrieved fragments revealed that: (*i*) small values of τ, namely, 1 and 2, cause the extraction of many fragments from one sentence corresponding to fragments expressed with different words in the other sentence: these are not ATFs. (*ii*) With $\tau = 3$, the heuristic is very precise and captures most ATFs appearing in the sentence pairs. (*iii*) Higher values of τ cause many valid fragments to be missed.

Once ATFs are generated, we need to label them as ancillary or important for PI such that this data can be used for training and testing ATC. Interestingly, the data can be automatically labeled exploiting the MSRP annotation: given a sentence pair from MSRP, we (*i*) extract the ATF from it and (*ii*) automatically annotate it as ancillary if the pair is a paraphrase and not ancillary otherwise. In other words, an ATF is considered ancillary only if it is extracted from a paraphrase pair. To verify the correctness of this approach, two experts manually labeled the obtained data extracted with $\tau = 3$ and found that only 3.3% of the data was mislabeled with respect to one annotator. The Cohen's kappa agreement between the annotators was 85%.

4 Experiments

In these experiments, we first evaluate state-of-the-art PI models to create our baseline, then we experiment with our ATC and finally, we combine them to show that ATC can improve PI.

4.1 Deriving PI baselines

Dataset. We used MSRP, which consists of 4,076 sentence pairs in the training set and 1,725 sentence pairs in test set. About 66% of the pairs are paraphrases. The pairs were extracted from topically similar Web news articles, applying some heuristics that select potential paraphrases to be annotated by human experts. We represent the sentence pairs using shallow trees generated with the Stanford parser[3].

Models. We adopted our state-of-the-art PI approach we proposed in (Filice et al., 2015). This, given two pairs of sentences, $p_a = \langle a_1, a_2 \rangle$ and $p_b = \langle b_1, b_2 \rangle$, represents instances as shown Fig. 1, and applies tree kernels to them. In particular, we used our best kernel com derived in the work above:

$$\begin{aligned}\text{SMK}(p_a, p_b) = \quad & sf\big(\text{SPTK}(a_1, b_1) \times \text{SPTK}(a_2, b_2), \\ & \text{SPTK}(a_1, b_2) \times \text{SPTK}(a_2, b_1)\big),\end{aligned}$$

where $sf(x_1, x_2) = \frac{1}{c} log(e^{cx_1} + e^{cx_2})$, and SPTK is the Smoothed Partial Tree Kernel (Croce et al., 2011). SMK considers the inherent symmetry of the PI task and evaluates the best alignment between the sentences in the input pairs. The sf is a softmax operation used in place of the max function[4], which is not a valid kernel function. SPTK uses a similarity function between words: we generated it with the word2vec tool[5] (Mikolov et al., 2013) using the

[3]`nlp.stanford.edu/software/corenlp.shtml`
[4]c=100 produces accurate approximation.
[5]`https://code.google.com/p/word2vec`

Model	Acc (%)	P	R	F1
LK	75.9	78.4	88.1	82.9
SMK	76.4	76.6	92.9	83.9
SMK+LK	**77.7**	79.4	8.99	**84.4**
(Socher et al., 2011)	76.8	–	–	83.6
(Madnani et al., 2012)	77.4	–	–	84.1

Table 2: Results on Paraphrase Identification.

skip-gram model applied to the UkWaC corpus (Baroni et al., 2009).

Results. As illustrated in Table 2, a binary Support Vector Machine equipped with SMK achieves a very high accuracy. Moreover, SMK combined with a linear kernel (LK) over similarity metrics[6] attains the state of the art in PI.

4.2 Experimental Evaluation on ATC

Dataset and models. We created an ATC dataset with $\tau = 3$ as described in Sec. 3. We make this dataset available[7]. We learned ATC with the C-SVM algorithm (Chang and Lin, 2011) inside KeLP[8]. The examples are represented using the shallow tree like the one on the right of Fig. 1. We used three different tree kernels: the Syntactic Tree Kernel (STK) by Collins and Duffy (2001), the Partial Tree Kernel (PTK) by Moschitti (2006) and SPTK using the word2vec similarity defined before.

Given the small size of such dataset (Only 8% of MSRP instances have additional fragments), we performed a 5-fold cross validation. Table 3 illustrates the Accuracy, Precision, Recall and F_1 of our models. ATC based on SPTK provides the best accuracy, i.e., 68.6%, which is a promising result for this research. The second most accurate classifier uses PTK, which is more flexible than STK.

4.3 Using ATC in PI

We carried out error analysis on PI and observed that the used classifier commits a systematic error: when two sentences share a very similar large part (identical in the extreme case) and one sentence has an ATF, it almost always classifies the sentences as

[6]These include *cosine similarities* of lemmas, POS-tags, and n-grams, *longest common substring* and *longest common subsequence* measures and Tree Kernel intra-pair similarities.

[7]`http://alt.qcri.org/resources/ancillary`

[8]`https://github.com/SAG-KeLP`

Kernel	Acc (%)	P	R	F_1
STK	65.1 ± 6.5	65.4 ± 8.0	58.3 ± 5.6	61.5 ± 5.8
PTK	67.4 ± 8.2	69.7 ± 8.8	56.5 ± 7.7	62.4 ± 7.8
SPTK	**68.6** ± 9.4	71.0 ± 9.0	57.9 ± 9.7	63.7 ± 9.3

Table 3: Results of Ancillary Text Classifiers

paraphrases, even if the ATF contains important information that invalidates the paraphrase relation. This kind of mistakes can be corrected by ATC.

Thus, we created the following ensemble model: given a pair to be classified, we apply our heuristic for ATF extraction. If the heuristic does not find any fragment in the pair, we only rely on the prediction provided by PI. Otherwise, we combine the prediction of ATC applied to ATF with the one of the PI classifier using a stacking strategy (Wolpert, 1992), i.e., the two predictions become the input features of a third classifier that makes the final decision.

To train this meta-classifier, we need the predictions from ATC and PI computed on a validation set. Hence, we split the training set in two parts: one part is used for training ATC and PI, while the other is classified with the trained models to produce the predictions for the meta-classifier. Then, the roles of the two parts are inverted. The meta-classifier is a linear SVM (Fan et al., 2008) implemented with KeLP.

Note that: (*i*) since we use 5-fold cross-validation, for each fold, we needed to apply the process described above to each fold; and (*ii*) all the learning algorithms and kernels adopt default parameters to also facilitate the reproducibility of our results.

Results. Table 4 reports the comparison between PI and PI combined with ATC (trained with SPTK). The performance is derived only on sentence pairs with ATFs.

The first column indicates the kernel used by the PI classifier, while the second column reports '+' or '-' to indicate if PI is combined with ATC or not, respectively. We note that ATC produces a great improvement, ranging from 8 absolute percent points over LK to about 3 points over SMK+LK, i.e., the state-of-the-art model. As expected, the more accurate the baseline is, the lower the improvement is produced.

It should be noted that only a relative small subset

PI	ATC	Acc (%)	P	R	F_1
LK	-	62.2 ± 5.4	57.8 ± 7.0	75.1 ± 7.4	65.3 ± 6.9
LK	+	70.4 ± 5.5†	69.2 ± 6.8	68.2 ± 4.7	68.7 ± 5.7
SMK	-	64.7 ± 6.0	59.0 ± 6.5	84.9 ± 4.6	69.5 ± 5.9
SMK	+	69.1 ± 5.5 ‡	66.9 ± 6.4	69.7 ± 5.8	68.2 ± 5.9
SMK+LK	-	70.5 ± 4.0	66.3 ± 6.7	77.3 ± 6.1	71.2 ± 5.3
SMK+LK	+	**73.2** ± 5.2‡	72.5 ± 5.4	71.3 ± 3.9	**71.8** ± 3.9

Table 4: PI classifier performance using ATC. The test set is restricted to examples having additional fragments. † and ‡ mark statistically significant differences in accuracy compared to the counterpart model not using ATC with confidence levels of 95% and 90%, respectively (t-test).

PI	ATC	Acc (%)	P	R	F_1
LK	-	75.5 ± 0.5	78.6 ± 0.9	87.6 ± 1.9	82.8 ± 0.4
LK	+	76.2 ± 1.0†	79.5 ± 0.1	87.2 ± 2.2	83.1 ± 0.8
SMK	-	75.6 ± 0.8	77.1 ± 0.4	90.7 ± 1.2	83.3 ± 0.7
SMK	+	75.9 ± 0.9†	77.9 ± 1.3	89.7 ± 1.2	83.4 ± 0.6
SMK+LK	-	78.1 ± 1.1	80.7 ± 0.6	88.6 ± 2.1	84.4 ± 0.9
SMK+LK	+	**78.3** ± 1.1‡	81.1 ± 0.9	88.2 ± 1.8	**84.5** ± 0.8

Table 5: PI classifier performance using ATC on the testset. † and ‡ mark statistically significant differences in accuracy compared to the counterpart model not using ATC with confidence levels of 95% and 90%, respectively (t-test).

of MSRP contains additional fragments (about 8% when $\tau = 3$). Thus, the impact on the entire PI testset cannot be large. Tab. 5 reports the accuracy of the previous models on the entire testset. An improvement over all models, state-of-the-art included, can be still observed, although it is less visible.

5 Conclusions

In this paper, we study and design models for learning to detect ancillary information in the context of PI. We used a heuristic rule for selecting additional fragments from paraphrase pairs, which, applied to MSRP, generates our ATF dataset. We manually annotated the latter for training and testing our ATCs.

Our experiments using several kernel models show that ATC can achieve a good accuracy (about 69%) and significantly impact the PI accuracy. Our results suggest that:

(*i*) it is possible to recognize information humans believe is ancillary; and

(*ii*) to go beyond the current results and technology for high-level semantic tasks (e.g., PI), we cannot just rely on shallow similarity features, but we rather need to build components that analyze different aspects of text and then combine the output of the different modules.

In the future, it would be interesting to use methods similar to those successfully used in question answering research, e.g., matching entities in the sentence trees using linked open data (Tymoshenko et al., 2014; Tymoshenko and Moschitti, 2015) or enriching trees with semantic information automatically produced by classifiers, e.g., (Severyn et al., 2013a; Severyn et al., 2013b).

Acknowledgements

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action) and by an IBM Faculty Award.

References

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transac-*

tions on Intelligent Systems and Technology, 2:27:1–27:27.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings EMNLP*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of COLING '04*, Stroudsburg, PA, USA.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural representations for learning relations between pairs of texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1003–1013, Beijing, China, July. Association for Computational Linguistics.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of NAACL HLT '12*. ACL.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proc. of ECML'06*, pages 318–329.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 741–750. ACM.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Building structures from classifiers for passage reranking. In *CIKM*.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning adaptable patterns for passage reranking. In *CoNLL*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 801–809.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *CIKM*, pages 1451–1460. ACM.

Kateryna Tymoshenko, Alessandro Moschitti, and Aliaksei Severyn. 2014. Encoding semantic resources in syntactic structures for passage reranking. In *Proceedings of EACL*.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.