Large-Scale Acquisition of Entailment Pattern Pairs by Exploiting Transitivity

Julien Kloetzer[*] **Kentaro Torisawa**[‡] **Chikara Hashimoto**[§] **Jong-Hoon Oh**[¶]
Information Analysis Laboratory,
National Institute of Information and Communications Technology (NICT), Kyoto, Japan
{[*]julien, [‡]torisawa, [§]ch, [¶]rovellia}@nict.go.jp

Abstract

We propose a novel method for acquiring entailment pairs of binary patterns on a large-scale. This method exploits the transitivity of entailment and a self-training scheme to improve the performance of an already strong supervised classifier for entailment, and unlike previous methods that exploit transitivity, it works on a large-scale. With it we acquired 138.1 million pattern pairs with 70% precision with such non-trivial lexical substitution as "*use Y to distribute X*"→"*X is available on Y*" whose extraction is considered difficult. This represents 50.4 million more pattern pairs (a 57.5% increase) than what our supervised baseline extracted at the same precision.

1 Introduction

Recognizing textual entailment (Geffet and Dagan, 2005; Androutsopoulos and Malakasiotis, 2009; Zanzotto et al., 2009; Berant et al., 2011) is an important task for many NLP applications, such as relation extraction (Romano et al., 2006) or question-answering (Harabagiu and Hickl, 2006). Text L ***entails*** text R if the information written in the latter can be deduced from the information written in the former. As building blocks to recognize entailment relations between texts, numerous works have focused on recognizing entailment relations between patterns, such as "*grew up in X*"→"*lived in X*" or "*X grew up in Y*"→"*X lived in Y*" (Lin and Pantel, 2001; Weeds and Weir, 2003a; Hashimoto et al., 2009; Berant et al., 2011; Kloetzer et al., 2013b).

We propose in this paper a method for acquiring on a very large-scale, entailment pairs of such class-dependent binary patterns as "*underwent* X_{exam} *on* Y_{date}"→"X_{exam} *carried out on* Y_{date}". Our starting point is a supervised baseline trained with 83,800 manually labeled pattern pairs detailed in Kloetzer et al. (2013b). Its top 205 million output pairs have an estimated 80% precision, but this baseline's performance is saturated. Table 1 shows the baseline's average precision when varying its amount of hand-labeled training data. Since the average precision only improves slightly with additional training data, the investment in hand-labeling additional training data is difficult to justify.

Quantity of training data	Average precision
Baseline plus 5,000 training data samples	**49.0%**
Baseline: 83,800 training data samples	48.8%
Baseline minus 10,000 training data samples	48.5%
Baseline minus 20,000 training data samples	47.8%

Table 1: Average precision for baseline method with various amounts of training data

To improve our baseline further, we exploit the transitivity property of entailment to automatically generate *new features* for it. The entailment is transitive; if we detect that L entails C and C entails R, we can infer an entailment relation between L and R even if no such relation was detected beforehand. Based on this idea, we propose a self-training scheme that works in the following way. For pattern pair $\langle P,Q\rangle$, we use the baselines output to find all the chains of patterns from P to Q that are linked by entailment relations, which we call *transitivity paths*, and encode the information related to them as new features to judge the validity of pair $\langle P,Q\rangle$. Our expectation is that even if our supervised baseline fails to judge $\langle P,Q\rangle$ as an entailment pair, the existence of paths from P to Q that are comprised of pairs judged as entailments by our baseline might strongly suggest that P entails Q; hence, adding our new features to the baseline should help it make better decisions based on the information encoded in the features. This self-training approach is the first that encodes the information contained in transitivity paths as features for a classifier, and as such it differs from previous state-of-the-art methods that exploit transitivity to extract new pairs using Integer Linear Programming (Berant et al., 2011) or that auto-generate training data (Kloetzer et al., 2013a).

From a corpus of 600 million web pages, we show that our proposed method extracted 217.8 million entailment pairs in Japanese with 80% precision[1], which is a 6% increase over the 205.3 million pairs output by our baseline with identical precision. It also extracted 138.1 million entailment pairs with 70% precision with non-trivial lexical substitution (generally deemed difficult to extract), which is a 50.4 million pair increase (57.5% size improvement) over the 87.7 million pairs output by our baseline with the same precision. These include such pairs as "*use X to distribute Y*"→"*Y is available on X*", "*underwent X on Y*"→"*X carried out on Y*", "*start X at Y*"→"*Y's X*" or "*attach X to Y*"→"*put X on Y*". Even though we only present results for the Japanese language, we believe that our method should be applicable to other languages as well. This is because none of the few language dependent features of our classifier are strictly needed by the baseline or our proposed method, and its performance boost is unrelated to these features.

2 Related Works

The task of recognizing entailment between texts has been proposed by Dagan et al. (2006) and intensively researched (Malakasiotis and Androutsopoulos, 2007; Szpektor et al., 2004; Androutsopoulos and Malakasiotis, 2009; Dagan et al., 2009; Hashimoto et al., 2009; Berant et al., 2011) using a various range of techniques, including Integer Linear Programming (Berant et al., 2011), machine learning with SVMs (Malakasiotis and Androutsopoulos, 2007), and probabilistic models (Wang and Manning, 2010; Shnarch et al., 2011). Entailment recognizer or entailment data sets have been used in such fields as relation extraction (Romano et al., 2006) and question-answering (Harabagiu and Hickl, 2006; Tanaka et al., 2013). In this work, we are interested into recognizing entailment between syntactic patterns, which can then be used as building blocks in a complete entailment recognition system (Shnarch et al., 2011). Recognizing entailment between patterns has generally been studied using unsupervised techniques (Szpektor et al., 2004; Hashimoto et al., 2009; Weeds and Weir, 2003b), although we showed that supervised techniques naturally obtain stronger performance (Kloetzer et al., 2013b).

The two works that are most closely related to our work are Berant et al. (2011) and Kloetzer et al. (2013a), both of which exploit transitivity to improve the result of a baseline classifier. Berant et al. (2011) proposed an entailment recognition method for binary patterns that exploits Integer Linear Programming techniques (ILP) to expand the results of an SVM classifier. This method encodes into an ILP problem an entailment graph, which is a valued graph where nodes and edges respectively represent patterns and their entailment relations, and the values equal the SVM classifiers score output. The problems variables $E_{PQ} \in \{0,1\}$ indicate whether pattern pairs (here, $\langle P,Q\rangle$) have an entailment relation, and the goal is to maximize the sum of the scores of the pairs selected as entailment relations $\{\langle P,Q\rangle | E_{PQ} = 1\}$. In (Kloetzer et al., 2013a), we proposed a contradiction acquisition method that uses a training data expansion scheme; it automatically generates new contradictions by exploiting transitivity and adds the highest scoring contradictions based on a novel score (CDP) to the training data of the original classifier. The score is based on the assumption that if pair $\langle P,Q\rangle$, when chained by transitivity to other pairs $\langle Q,R_i\rangle$, generally leads to correct entailment pairs $\langle P,R_i\rangle$, then *all* pairs $\langle P,R_i\rangle$ should be correct entailment pairs. Although this work was designed for contradiction recognition, it is easily adapted to entailment.

3 Target Data and Baseline Classifiers

Target Pattern Pairs We extracted our binary patterns from the TSUBAKI corpus (Shinzato et al., 2008) of 600 million Japanese web pages. Binary patterns are defined as sequences of words on the path of dependency relations connecting two nouns in a sentence and have two variables. "*use Y to distribute X*" and "*X is available on Y*" are such

[1]Examples are given in English for convenience

binary patterns. Like previous works (De Saeger et al., 2009; Berant et al., 2011; Kloetzer et al., 2013a), we pose restrictions on the noun-pairs that co-occur with each pattern using word classes to disambiguate their various potential meanings: "X_{book} *by* Y_{author}" and "$X_{building}$ *by* $Y_{location}$". We used the EM-based noun clustering algorithm presented inKazama and Torisawa (2008) to classify one million nouns into 500 semantic classes. Our target set, to which we apply all of our classifiers, is set Σ of around 11 billion class-dependent pattern pairs for which both patterns share at least three co-occurring noun-pairs.

Baseline Classifier Our baseline classifier (**BASE**) is an SVM classifier trained with about 83,800 binary pattern pairs that were hand-labeled as entailment (25,436 pairs, 30.4% of the total) or non-entailment (58,361 pairs). We trained the classifier using SVMlight software[2] with a polynomial kernel of degree 2.

Following previous work (Kloetzer et al., 2013b), we used three types of features in **BASE**: **surface features** indicate clues like the presence of n-grams or measure the string overlap between two patterns; **database features** exploit existing language resources; and **distributional similarity scores** measure the patterns' semantic similarity based on the nouns that co-occur with them. See Kloetzer et al. (2013b) for more details about **BASE** s features.

4 Proposed Method

Our method consists of the following three steps:

Step 1 Chain together the entailment pairs provided by our baseline classifier **BASE** to form transitivity paths; if $P \to Q$ and $Q \to R$, then create path $P \to Q \to R$.

Step 2 Train new classifiers with features that encode the information contained in the transitivity paths obtained in Step 1.

Step 3 Combine the output of these classifiers with that of baseline classifier **BASE**.

Figure 1 shows an overview of our method, and we describe its details in the following sections.

[2]http://svmlight.joachims.org/

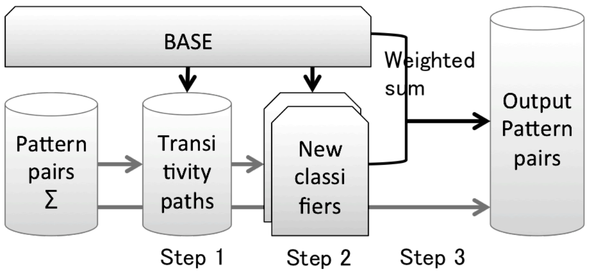

Figure 1: Overview of proposed method

4.1 Step 1: Transitivity Paths

We generate chains of entailment pairs (or transitivity paths) in the following way. First, we extract from the output of the baseline classifier **BASE** set $E(\theta)$ of the pattern pairs for which **BASE** returns a score over given threshold θ: $E(\theta) = \{\langle P, Q\rangle \in \Sigma | S_{BASE}(P, Q) \geq \theta\}$, where $S_{BASE}(P, Q)$ is the score returned by **BASE** for pattern pair $\langle P,Q\rangle$. The higher θ is, the greater the precision of the pairs in $E(\theta)$ should be. Then by chaining the entailment pairs from $E(\theta)$ together, we build sets of transitivity paths composed of two entailment pairs $Tr(\theta, 1)$ for $\theta \in \{0, -\infty\}$ and of three entailment pairs $Tr(\theta, 2)$ for $\theta = 0$. Since additional chaining is computationally expensive, we stopped at paths that consist of three pairs.

4.2 Step 2: Training New Classifiers

In this step, we train new classifiers by adding new features to **BASE**. The training data, the classifier software, and the settings are the same as for **BASE**. For given pattern pair $\langle P,R\rangle$, $Path(P, R, \theta, N)$ is the set of all the transitivity paths in $Tr(\theta, N)$ that lead to pair $\langle P,R\rangle$. We encode the information contained in these paths in three new feature sets.

Before explaining these three new feature sets, we define three scoring functions for the transitivity paths to assess their quality; the *MinScore* of a path is the minimum of scores returned by **BASE** for each pair in the path, and *ArScore* and *GeoScore* are the arithmetic and geometric averages of the scores returned by **BASE** for each pair in the path. Each of the three feature sets is computed for each of the three scoring functions, but we just mention *MinScore* in our explanations due to space limitations.

Feature set 1: scores of top-ranked paths Here we select the top ten paths of $Path(P, R, \theta, N)$ ranked by *MinScore* and use as features a new vector that consists of the following values: **(1)** the *MinScore* of each path and **(2)** the scores returned by **BASE** for each of the pairs in the ten paths. When there are fewer than ten paths, the missing features are set to 0.

Feature set 2: BASE features of the pairs in the highest ranking transitivity paths Here we select the transitivity path of $Path(P, R, \theta, N)$ with the highest *MinScore* and use the **BASE** feature values for each pair in the transitivity path as a new feature vector for pair $\langle P,R \rangle$.

Feature set 3: score distribution Given threshold α, we count the number of paths whose *MinScore* exceeds α. By varying α from lower bound low to upper bound up, we derive vector $[|\{p \in Path(P, R, \theta, N) | MinScore(p) \geq \alpha\}|]_{\alpha \in \{low, low+\beta, low+2*\beta, ..., up\}}$ and use it as a new feature vector for pair $\langle P,R \rangle$. We set $\beta = 0.1$ and low and up such that the score values returned by **BASE** are bounded by low and up.

4.3 Step 3: Optimization and Weighted Sum Classifier Combination

The final output of our method combines the outputs of **BASE** and two new classifiers: **(1)** a classifier with new features computed with 1-step transitivity paths ($N = 1$), and **(2)** another with new features computed with up to 2-step transitivity paths ($N \in \{1, 2\}$). We then use a weighted sum and compute score $S_{PROPOSED}(P, Q) = \sum_i n_i * S_i(P, Q)$ for each pair $\langle P,Q \rangle$. S_i represents the scores of the respective classifiers, and we set $n_0 + n_1 + n_2 = 100$ (n_i are all natural numbers).

For each potential combination of three weights n_i, we computed the average precision returned by our method on **DEV**, our development set, and selected for our final output the weight combination that gave the best average precision on **DEV**. The final classifiers weights obtained in our experiments were 62 for **BASE**, 30 for the classifier with 1-step transitivity features, and 8 for the one with the 1- and 2-step transitivity features.

Using the same method, we also performed ablation tests to remove the features that harmed the classifiers and ensured that every proposed new feature set and every scoring function were useful.

Finally, we confirmed that using a weighted sum for our proposed method returned higher average precision than Stacking (Wolpert, 1992),

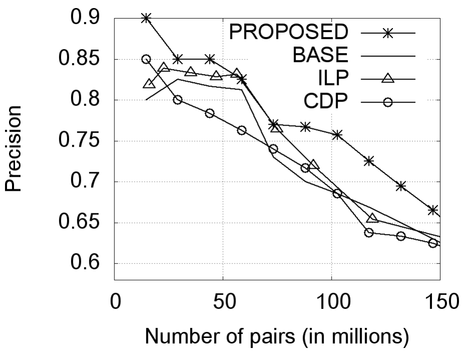

Figure 2: Precision curves for **PROPOSED** and baseline methods for *non-similar* pairs

which is a more standard combination method, or than using the output of any of the new classifiers alone.

5 Experiments

In this section, we evaluate our proposed method in a large-scale setting and compare it to **BASE** and to state-of-the-art methods based on ILP (Berant et al., 2011) and automatic training data expansion (Kloetzer et al., 2013a). We also indicate that our method shows the best performance gain for pattern pairs with non-trivial lexical substitutions, which are more difficult to acquire.

Evaluation Method For our evaluation, we prepared test set **TEST** of 15,000 pattern pairs randomly sampled from Σ (our target set of 11 billion pairs). The pairs were annotated by three humans (not the authors) who voted to settle labeling discrepancies. We also prepared development set **DEV** of $5,000$ pattern pairs from Σ for tuning our method. The Kappa score was 0.55 for the annotation of these two sets.

We measured the performance of each method by computing its *average precision* (Manning and Schütze, 1999) on the **TEST** set. We used the average precision instead of the traditional F-value because the latters value greatly varies depending on where the classification boundary is drawn, even for similar rankings. We also drew precision curves for each method using the same **TEST** set.

Proposed Methods Performance We first show the performance of **PROPOSED** (our proposed method) and **BASE** (the baseline classifier). As another baseline, we consider **BASE +DEV** where the 5,000 samples of the **DEV** set were added to the **BASE** training data. We show the average precision for each of these three classifiers and the

Classifier	Average precision	Million pairs at 80% prec.
PROPOSED	**50.64%**	**217.8**
BASE + DEV	48.96%	202.4
BASE	48.79%	205.3
ILP	N/A	205.2
CDP	48.42%	198.0

Table 2: Average precision and entailment pairs obtained (in millions) for proposed method, baseline classifiers, and state-of-the-art methods

Classifier	Av. precision (*similar*)	Av. precision (*non-similar*)
PROPOSED	**78.73%**	**39.53%**
BASE + DEV	77.72%	37.24%
BASE	77.85%	36.98%

Table 3: Average precision for *similar* and *non-similar* pairs

number of pairs obtained at 80% precision in Table 2. We also show the performance of these classifiers over *similar* pattern pairs (both patterns share a content word) and *non-similar* pairs (they do not share a content word) in Table 3.

As mentioned in the introduction, **BASE +DEV** shows that the addition of 5,000 hand-labeled samples to the training data of **BASE** (a 6% increase) only improves the average precision performance by 0.17%. Our proposed method, on the other hand, exploits the same 5,000 new annotated samples for tuning its parameters and obtains a 1.85% gain of average precision. Using **PROPOSED**, we acquired 217.8 million pattern pairs with 80% precision, an improvement of 6.0% over **BASE**.

As shown in Table 3, **BASE** s performance is much lower for *non-similar* pairs like "*use Y to distribute X*"→"*X is available on Y*", which have non-trivial lexical substitutions and are more difficult to acquire than *similar* pairs. This is also where **PROPOSED** obtains the biggest gain in performance: an average precision of 39.53 compared to 36.98 for **BASE**. We show the precision curves we obtained when ranking the *non-similar* pairs with **BASE** and **PROPOSED** in Fig. 2. **PROPOSED** acquired 138.1 million *non-similar* pairs at 70% precision, which is an increase of 50.4 million pairs (a 57.5% size improvement) compared to **BASE** with the same precision. We believe that the strong performance of **BASE** for *similar* entailment pairs helped it discover, through transitivity, the variations of *non-similar* entailment pairs it could already detect.

Comparison to State-of-the-art Methods We also compared **PROPOSED** with two state-of-the-art methods that exploit transitivity: the ILP-based method of Berant et al. (2011) (**ILP**) and the training data expansion method we proposed in Kloetzer et al. (2013a) (**CDP**). The latter, which was initially designed for acquiring contradiction pairs, was adapted to acquire entailment for comparison purposes. The results of this comparison are summarized in Table 2, and the precision curves for these two methods as well over *non-similar* pairs are shown in Fig. 2. Our proposed method is the only one that provides stable improvement in our large-scale setting; at best, the other two just slightly outperform **BASE**. We believe that our feature encoding provides more information to the classifier than the raw scores in the transitivity paths that are exploited by the other state-of-the-art methods, and as such strengthens the performance.

As for explaining the poor performance of the state-of-the-art methods, ILP is unfortunately not tractable for big problems; our ILP solver failed to solve 82% of the independent problems we fed it due to insufficient memory even on 64-Gb memory machines, making **ILP** just slightly better than **BASE**. Our pattern graph is also sparser than that in Berant et al. (2011), and as such ILP might not be completely efficient. But we assume that even if we had used the graphs whole closure, the ILP problem instances would have become even less tractable, resulting in performance that only slightly exceeds **BASE**. As for **CDP**, since our baseline classifier already has more than 80,000 hand-annotated samples as training data, the addition of automatically generated training samples is actually harmful.

6 Conclusion

In this work, we proposed a method that exploits the transitivity relation of entailment in a self-training scheme and combines classifiers with a weighted sum. In our large-scale setting, our method outperforms state-of-the-art methods that are also based on a transitivity approach, including an ILP-based method. Using our proposed method, we acquired 217.8 million Japanese entailment pairs with 80% precision and 138.1 million non-trivial pairs with 70% precision. We are considering an extrinsic evaluation for these data such as the RTE test in future research.

References

Ion Androutsopoulos and Prodromos Malakasiotis. 2009. A survey of paraphrasing and textual entailment methods. *arXiv preprint arXiv:0912.3747*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL 2011*, pages 610–619.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):1–17.

Stijn De Saeger, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proceedings of ICDM 2009*, pages 764–769.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 107–114. Association for Computational Linguistics.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912. Association for Computational Linguistics.

Chikara Hashimoto, Kentaro Torisawa, Kow Kuroda, Stijn De Saeger, Masaki Murata, and Jun'ichi Kazama. 2009. Large-scale verb entailment acquisition from the web. In *Proceedings of EMNLP 2009*, volume 3, pages 1172–1181.

Junichi Kazama and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. *Proceedings of ACL 2008*, pages 407–415.

Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, Kiyonori Ohtake, and Motoki Sano. 2013a. Two-stage method for large-scale acquisition of contradiction pattern pairs using entailment. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 693–703.

Julien Kloetzer, Stijn De Saeger, Kentaro Torisawa, Motoki Sano, Chikara Hashimoto, and Jun Gotoh. 2013b. Large-scale acquisition of entailment pattern pairs. In *Information Processing Society of Japan (IPSJ) Kansai-Branch Convention*.

Dekang Lin and Patrick Pantel. 2001. Dirt - discovery of inference rules from text. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using SVMs and string similarity measures. In *Proceedings of the ACL- PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47.

Christopher D Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT press.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, and Ido Dagan. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proceedings of EACL*, pages 409–416.

Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. TSUBAKI: an open search engine infrastructure for developing new information access methodology. *Proceedings of IJCNLP2008*.

Eyal Shnarch, Jacob Goldberger, and Ido Dagan. 2011. A probabilistic modeling framework for lexical entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 558–563, Stroudsburg, PA, USA. Association for Computational Linguistics.

Idan Szpektor, Hristo Tanev, Ido Dagan, Bonaventura Coppola, et al. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP*, volume 4, pages 41–48.

Masahiro Tanaka, Stijn De Saeger, Kiyonori Ohtake, Chikara Hashimoto, Makoto Hijiya, Hideaki Fujii, and Kentaro Torisawa. 2013. Wisdom2013: A large-scale web information analysis system. In *Sixth International Joint Conference on Natural Language Processing*, pages 58–61.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1164–1172, Beijing, China. Association for Computational Linguistics. ACM ID: 1873912.

Julie Weeds and David Weir. 2003a. A general framework for distributional similarity. In *Proceedings of EMNLP 2003*, pages 81–88. Association for Computational Linguistics.

Julie Weeds and David Weir. 2003b. A general framework for distributional similarity. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 81–88. Association for Computational Linguistics.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

Fabio Massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Natural Language Engineering*, 15(04):551–582.