Comparing Word Representations for Implicit Discourse Relation Classification

Chloé Braud
ALPAGE, Univ Paris Diderot
& INRIA Paris-Rocquencourt
75013 Paris - France
chloe.braud@inria.fr

Pascal Denis
MAGNET, INRIA Lille Nord-Europe
59650 Villeneuve d'Ascq - France
pascal.denis@inria.fr

Abstract

This paper presents a detailed comparative framework for assessing the usefulness of unsupervised word representations for identifying so-called implicit discourse relations. Specifically, we compare standard one-hot word pair representations against low-dimensional ones based on Brown clusters and word embeddings. We also consider various word vector combination schemes for deriving discourse segment representations from word vectors, and compare representations based either on all words or limited to head words. Our main finding is that denser representations systematically outperform sparser ones and give state-of-the-art performance or above without the need for additional hand-crafted features.

1 Introduction

Identifying discourse relations is an important task, either to build a discourse parser or to help other NLP systems such as text summarization or question-answering. This task is relatively straightforward when a discourse connective, such as but or because, is used (Pitler and Nenkova, 2009). The identification becomes much more challenging when such an overt marker is lacking, and the relation needs to be inferred through other means. In (1), the presence of the pair of verbs (rose,tumbled) triggers a *Contrast* relation. Such relations are extremely pervasive in real text corpora: they account for about 50% of all relations in the Penn Discourse Treebank (Prasad et al., 2008).

(1) [Quarterly revenue **rose** 4.5%, to $2.3 billion from $2.2 billion]$_{arg1}$ [For the year, net income **tumbled** 61% to $86 million, or $1.55 a share]$_{arg2}$

Automatically classifying implicit relations is difficult in large part because it relies on numerous factors, ranging from syntax, and tense and aspect, to lexical semantics and even world knowledge (Asher and Lascarides, 2003). Consequently, a lot of previous work on this problem have attempted to incorporate some of these information into their systems. These assume the existence of syntactic parsers and lexical databases of various kinds, which are available but for a few languages, and they often involve heavy feature engineering (Pitler et al., 2009; Park and Cardie, 2012). While acknowledging this knowledge bottleneck, this paper focuses on trying to predict implicit relations based on easily accessible lexical features, targeting in particular simple word-based features, such as pairs like (rose,tumbled) in (1).

Most previous studies on implicit relations, going back to (Marcu and Echihabi, 2002), incorporate word-based information in the form of word pair features defined across the pair of text segments to be related. Such word pairs are often encoded in a *one-hot* representation, in which each possible word pair corresponds to a single component of a very high-dimensional vector. From a machine learning perspective, this type of sparse representation makes parameter estimation extremely difficult and prone to overfitting. It also makes it difficult to achieve any interesting semantic generalization. To see this, consider the distance (e.g., Euclidean or cosine) induced by such representation. Assuming for simplicity that one characterizes each pair of discourse segments via their main verbs, the corresponding one-hot encoding for the pair (rose,tumbled) would be at equal distance from the synonymic pair (went_up,lost) and the antonymic pair (went_down,gained), as all three vectors are orthogonal to each others.

Various attempts have been made at reducing sparsity of lexical features. Recently, Ruther-

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2201–2211,
Lisbon, Portugal, 17-21 September 2015.

ford and Xue (2014) proposed to use Brown clusters (Brown et al., 1992) for this task, in effect replacing each token by its cluster binary code. These authors conclude that these denser, cluster-derived representations significantly improve the identification of implicit discourse relations and report the best performance to date using also additional features. Unfortunately, their claim is somewhat weakened by the fact that they fail to compare the use of their cluster word pairs against other types of word representations, including one-hot encodings of word pairs or other low-dimensional word representations. This work also leaves other important questions open. In particular, it is unclear whether all word pairs constructed over the two discourse segments are truly informative and should be included in the model. Given that word embeddings capture latent syntactic and semantic information, yet another important question is to which extent the use of these representations dispenses us from using additional hand-crafted syntactic and semantic features.

This paper fills these gaps and significantly extends the work of (Rutherford and Xue, 2014) by explicitly comparing various types of word representations and vector composition methods. Specifically, we investigate three well-known word embeddings, namely Collobert and Weston (Collobert and Weston, 2008), hierarchical log-bilinear model (Mnih and Hinton, 2007) and Hellinger Principal Component Analysis (Lebret and Collobert, 2014), in addition to Brown cluster-based and standard one-hot representations. All these word representations are publicly available for English and can be easily acquired for other languages just using raw text data, thus alleviating the need for hand-crafted lexical databases. This makes our approach easily extendable to resource-poor languages. In addition, we also investigate the issue of which specific words need to be fed to the model, by comparing using just pairs of verbs against all pairs of words, and how word representations should be combined over discourse segments, comparing component-wise product against simple vector concatenation.

2 Word Representations

A word representation associates a word to a mathematical object, typically a high-dimensional vector in $\{0,1\}^{|\mathcal{V}|}$ or $\mathbb{R}^{|\mathcal{V}|}$, where $\mathcal{V}$ is a base vocabulary. Each dimension of this vector corresponds to a feature which might have a syntactic or semantic interpretation. In the following, we review different types of word representations used in NLP.

2.1 One-hot Word Representations

Given a particular NLP problem, the crudest and yet most common type of word representation consists in mapping each word into a one-hot vector, wherein each observed word corresponds to a distinct vector component. More formally, let $\mathcal{V}$ denote the set of all words found in the texts and w a particular word in $\mathcal{V}$. The one-hot representation of w is the d-dimensional indicator vector, noted $\mathbb{1}_w$, such that $d = |\mathcal{V}|$: that is, all of this vector's components are 0's but for one 1 component corresponding to the word's index in $\mathcal{V}$. It is easy to see that this representation is extremely sparse, and makes learning difficult as it mechanically blows up the parameter space of the model.

2.2 Clustering-based Word Representations

An alternative to these very sparse representations consists in learning word representations in an unsupervised fashion using clustering. An example of this approach are the so-called *Brown clusters* induced using the Brown hierarchical clustering algorithm (Brown et al., 1992) with the goal of maximizing the mutual information of bigrams. As a result, each word is associated to a binary code corresponding to the cluster it belongs to. Given the hierarchical nature of the algorithm, one can create word classes of different levels of granularity, corresponding to bit codes of different sizes. The less clusters, the less fine-grained the distinctions between words but the less sparsity. Note that this kind of representations also yields one-hot encodings but on a much smaller vocabulary size (i.e., the number of clusters). Brown clusters have been used for several NLP tasks, including NER, chunking (Turian et al., 2010), parsing (Koo et al., 2008) and implicit discourse relation classification (Rutherford and Xue, 2014).

2.3 Dense Real-Valued Representations

Another approach to induce word representations from raw text is to learn *distributed* word representations (aka word embeddings), which are dense, low-dimensional, and real-valued vectors. These are typically learned using neural language models (Bengio et al., 2003). Each dimension corresponding to a latent feature of the word that captures paradigmatic information. An example of such

embeddings are the so-called Collobert and Weston embeddings (Collobert and Weston, 2008). The embeddings are learned discriminatively by minimizing a loss between the current n-gram and a corrupted n-gram whose last word comes from the same vocabulary but is different from the last word of the original n-gram. Another example are the Hierarchical log-bilinear embeddings (Mnih and Hinton, 2007) induced using a probabilistic and linear neural model, with a hierarchical principle used to speed up the model evaluation. The embeddings are obtained by concatenating the embeddings of the $n-1$ words of a n-gram and learning the embedding of the last word.

A final approach is based on the assumption that words occurring in similar contexts tend to have similar meanings. Building word *distributional* representations is done by computing the raw cooccurrence frequencies between each word and the $|\mathcal{D}|$ words that serve as context, with $\mathcal{D}$ generally smaller than the overall vocabulary, then applying some transformation (e.g. TF-IDF). As $|\mathcal{D}|$ is generally too large to form a tractable representation, a dimensionality reduction algorithm is used to end up with $p \ll |\mathcal{D}|$ dimensions. Like for distributed representations, we end up with a dense low-dimensional real-valued vector for each word. A recent example of such approach is the Hellinger PCA embeddings of (Lebret and Collobert, 2014) which were built using Principal Component Analysis based on Hellinger distance as dimensionality reduction algorithm. An important appeal of these representations is that they are much less time-consuming to train than the ones based on neural language models while allowing similar performance (Lebret and Collobert, 2014).

3 Segment Pair Representations

We now turn to the issue of combining the word representations as described in section 2 into composite vectors corresponding to implicit discourse classification instances. Schematically, the representations employed for pairs of discourse segments differ along three main dimensions. First, we compare the use of a single word per segment (roughly, the two main verbs) against that of all the words contained in the two segments. Second, we compare the use of sparse (i.e., one-hot) vs. dense representations for words. As discussed, Brown cluster bit representations are a special (i.e., low-dimensional) version of one-hot encoding. Third, we use two different types of combinations of word vectors to yield segment vector representations: concatenation and Kronecker product. The proposed framework is therefore much more general than the one given in previous work such as (Rutherford and Xue, 2014).

3.1 Notation

Our classification inputs are pairs of text segments, the two arguments of the relation to be predicted. Let $S_1 = \{w_{1_1}, \ldots, w_{1_n}\}$ denote the n words that make up the first segment and $S_2 = \{w_{2_1}, \ldots, w_{2_m}\}$ the m words in the second segment. That is, we regard segments as bags of words. Let $\mathcal{V}$ again denote the word vocabulary, that is the set of all words found in the segments. Sometimes, we will find it useful to refer to a particular subset of $\mathcal{V}$. Let $\mathrm{head}(\cdot)$ refer to the function that extracts the head word of segment S,[1] and $\mathcal{V}_h \subseteq \mathcal{V}$ the set of head words. As our goal is to compare different feature representations, we define Φ as a generic feature function mapping pairs of segments to a d-dimensional real vector:

$$\begin{aligned}\Phi: \ & \mathcal{V}^n \times \mathcal{V}^m \rightarrow \mathbb{R}^d \\ & (S_1, S_2) \mapsto \Phi(S_1, S_2)\end{aligned}$$

The goal of learning is to acquire for each relation a linear classification function $f_{\boldsymbol{w}}(\cdot)$, parametrized by $\boldsymbol{w} \in \mathbb{R}^d$, mapping $\Phi(S_1, S_2)$ into $\{-1, +1\}$.

Recall that $\mathbb{1}_w$ refers to the d-dimensional one-hot encoding for word $w \in \mathcal{V}$. Let us also denote by $\oplus$ and $\otimes$ the vector concatenation operator and the Kronecker product, respectively. Note that doing a Kronecker product on two vectors $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$ is equivalent to doing the outer product $\boldsymbol{u}\boldsymbol{v}^\top \in \mathbb{R}^{m \times n}$. Finally, the $\mathrm{vec}(\cdot)$ operator converts a $m \times n$ matrix into an $mn \times 1$ column vector by stacking its columns.

3.2 Representations based on head words

One of the simplest representation one can construct for a pair of segments (S_1, S_2) is to consider only their head words: $h_1 = \mathrm{head}(S_1)$ and $h_2 = \mathrm{head}(S_2)$. In this simple scenario, two main questions that remain are: (i) which vector representations do we use for h_1 and h_2, and (ii) how do we combine these representations. An important criterion for word vector combination is that they retain text ordering information between text segments which really matters for this task.

[1] Head extraction will be detailed in section 4.1.

Thus, inverting the order between two main verbs (e.g., push and fall) will often lead to distinct discourse relation being inferred, as some relations are asymmetric (e.g., *Result* or *Explanation*).

One-hot representations Starting again with the simplest case, one can use the one-hot encodings corresponding to the two head words, $\mathbb{1}_{h_1}$ and $\mathbb{1}_{h_2}$ respectively, and combine them using either concatenation or product, leading to our two first feature mappings:

$$\Phi_{h,\mathbb{1},\oplus}(S_1, S_2) = \mathbb{1}_{h_1} \oplus \mathbb{1}_{h_2}$$
$$\Phi_{h,\mathbb{1},\otimes}(S_1, S_2) = \text{vec}(\mathbb{1}_{h_1} \otimes \mathbb{1}_{h_2})$$

Note that $\Phi_{h,\mathbb{1},\oplus}(S_1, S_2)$ lives in $\{0,1\}^{2|\mathcal{V}_h|}$ and $\Phi_{h,\mathbb{1},\otimes}$ in $\{0,1\}^{|\mathcal{V}_h|^2}$. The latter representation amounts to assigning one 1 component for each pair of words in $\mathcal{V}_h \times \mathcal{V}_h$, and is the sparsest representation one can construct from head words alone. In some sense, it is also the most expressive in that we learn one parameter for each word pair, hence capturing interaction between words across segments. By contrast, $\Phi_{h,\mathbb{1},\oplus}(S_1, S_2)$ doesn't explicitly model word interaction across discourse segments, treating each word in a given segment (left or right) as a separate dimension.

Dense representations Alternatively, one can represent head words through their real low-dimensional embeddings. Let $\boldsymbol{M}$ denote a $n \times p$ real matrix, wherein the i^{th} row corresponds to the p-dimensional embedding of the i^{th} word of $\mathcal{V}_h$, with $p \ll |\mathcal{V}_h|$.[2] Using this notation, one can derive the word embeddings of the head words h_1 and h_2 from their one-hot representations using simple matrix multiplication: $\boldsymbol{M}^\top \mathbb{1}_{h_1}$ and $\boldsymbol{M}^\top \mathbb{1}_{h_2}$, respectively. Concatenation and product yield two new feature mappings, respectively:

$$\Phi_{h,\boldsymbol{M},\oplus}(S_1, S_2) = \boldsymbol{M}^\top \mathbb{1}_{h_1} \oplus \boldsymbol{M}^\top \mathbb{1}_{h_2}$$
$$\Phi_{h,\boldsymbol{M},\otimes}(S_1, S_2) = \text{vec}(\boldsymbol{M}^\top \mathbb{1}_{h_1} \otimes \boldsymbol{M}^\top \mathbb{1}_{h_2})$$

These new representations live in a much lower dimensional real spaces: $\Phi_{h,\boldsymbol{M},\oplus}(S_1, S_2)$ lives in $\mathbb{R}^{2p}$ and $\Phi_{h,\boldsymbol{M},\otimes}(S_1, S_2)$ in $\mathbb{R}^{p^2}$.

3.3 Representations based on all words

The various segment-pair representations that we derived from pairs of head words can be generalized to the case in which we keep all the words in each segment. The additional issue in this context is in the combination of the different word vector representations within and across the two segments, and that of normalizing the segment vectors thus obtained. For simplicity, we assume that the representation for each segment is computed by summing over the pairs of words vectors composing the segments.

One-hot representations Following this approach and recalling that S_1 contains n words, while S_2 has m words, one can construct one-hot encodings for segment pairs as follows:

$$\Phi_{all,\mathbb{1},\oplus}(S_1, S_2) = \sum_i^n \sum_j^m \mathbb{1}_{w_{1_i}} \oplus \mathbb{1}_{w_{2_j}}$$
$$\Phi_{all,\mathbb{1},\otimes}(S_1, S_2) = \sum_i^n \sum_j^m \text{vec}(\mathbb{1}_{w_{1_i}} \otimes \mathbb{1}_{w_{2_j}})$$

If used without any type of frequency thresholding, these mappings result in very high-dimensional feature representations living in $\mathbb{Z}_{\geq 0}^{2|\mathcal{V}|}$ and $\mathbb{Z}_{\geq 0}^{|\mathcal{V}|^2}$, respectively. Interestingly, note that the feature mapping $\Phi_{all,\mathbb{1},\otimes}(S_1, S_2)$ corresponds to the standard segment-pair representation used in many previous work, as (Marcu and Echihabi, 2002; Park and Cardie, 2012).

Dense representations We can apply the same composition operations to denser representations, yielding two new mappings:

$$\Phi_{all,\boldsymbol{M},\oplus}(S_1, S_2) = \sum_{i,j}^{n,m} \boldsymbol{M}^\top \mathbb{1}_{w_{1_i}} \oplus \boldsymbol{M}^\top \mathbb{1}_{w_{2_j}}$$
$$\Phi_{all,\boldsymbol{M},\otimes}(S_1, S_2) = \sum_{i,j}^{n,m} \text{vec}(\boldsymbol{M}^\top \mathbb{1}_{w_{1_i}} \otimes \boldsymbol{M}^\top \mathbb{1}_{w_{2_j}})$$

Like their head word versions, these vectors live in $\mathbb{R}^{2p}$ and $\mathbb{R}^{p^2}$, respectively.

Vector Normalization Normalization is important as unnormalized composite vectors are sensitive to the number of words present in the segments. The first type of normalization we consider is to simply convert our vector representation into vectors on the unit hypersphere: this is achieved by dividing each vector by its L_2 norm.

Another type of normalization is obtained by inverting the order of summation and concatenation in the construction of composite vectors. Instead

[2] For now, we assume that $n = \mathcal{V}_h$ which is unrealistic. See section 4.1 for a discussion of unknown words.

of summing over concatenated pairs of word vectors across the two segments, one can first sum individual word vectors within each segment, then concatenate the two segment vectors. One can thus use mapping $\Phi'_{all,\mathbb{1},\oplus}$ in lieu of $\Phi_{all,\mathbb{1},\oplus}$:

$$\Phi'_{all,\mathbb{1},\oplus}(S_1, S_2) = \sum_i^n \mathbb{1}_{w_{1_i}} \oplus \sum_j^m \mathbb{1}_{w_{2_j}}$$

It should be clear that $\Phi'_{all,\mathbb{1},\oplus}$ provides a normalized version of $\Phi_{all,\mathbb{1},\oplus}$ as this latter mapping amounts to weighted versions of the former:

$$\Phi_{all,\mathbb{1},\oplus}(S_1, S_2) = m \sum_i^n \mathbb{1}_{w_{1_i}} \oplus n \sum_j^m \mathbb{1}_{w_{2_j}}$$

4 Experiment Settings

Through the comparative framework described in section 3, our objective is to assess the usefulness of different vectorial representations for pairs of discourse segments. Specifically, we want to establish whether dense representations are better than sparse ones, and whether certain word pairs are more relevant than others, which resource and which combination schemes are more adapted to the task, and, finally, whether standard features derived from external databases are still relevant in the presence of dense representations.

4.1 Feature Set

Our main features are primarily lexical in nature and based on surface word forms. These are defined either on *all* words used in the relation arguments or only on their heads.

Head Extraction Heads of discourse segments are first extracted using Collins syntactic head rules[3]. In order to retrieve the semantic predicate, we define a heuristic which looks for the past participle of auxiliaries, the adjectival or nominal attribute of copula, the infinitive complementing "have to" forms and the first head of coordination conjunctions. In case of multiple subtrees, we look for the head of the first independent clause, or, failing that, of the first phrase.

Word Representations We use either one-hot encodings or use word embeddings to build denser representations as described in section 3. The Brown clusters (*Brown*), Collobert-Weston (*CnW*) representations, and the hierarchical log-bilinear (*HLBL*) embeddings correspond to the versions implemented in (Turian et al., 2010)[4]. They have been built on Reuters English newswire with case left intact. We test versions with 100, 320, 1000 and 3,200 clusters for *Brown*, with 25, 50, 100 and 200 dimensions for *CnW* and with 50 and 100 dimensions for *HLBL*. The Hellinger PCA (*H-PCA*) embeddings come from (Lebret and Collobert, 2014)[5] and have been built over the entire English Wikipedia, the Reuters corpus and the Wall Street Journal with all words in lower case. The vocabulary corresponds to the words that appear at least 100 times and normalized frequency is computed with the 10,000 most frequent words as context words. We test versions with 50, 100 and 200 dimensions for *H-PCA*. The coverage of each resource is presented in table 1.

	# words	# missing words	
		All words	Head words
HLBL	246,122	5,439	171
CnW	268,810	5,638	171
Brown	247,339	5,413	171
H-PCA	178,080	7,042	190

Table 1: Word embeddings and Brown clusters lexicon coverage.

When presenting our results, we distinguish between systems based on one-hot encoding built from raw tokens (*one-hot*) or Brown clusters (*Brown*). We group the systems that use embeddings under *Embed.* When relevant, we indicate the number of dimensions (e.g. *Brown 3,200* is the system using Brown clusters with 3,200 clusters). We use the symbols defined in section 3 to represent the operation linking the arguments representations (e.g. *one-hot* $\oplus$ corresponds to the transformation defined by $\Phi_{h,\mathbb{1},\oplus}$ when using heads and by $\Phi_{all,\mathbb{1},\oplus}$ when using all words).

Vocabulary Sizes For one-hot encoding, the case is left intact. We ignore the unknown words when using the Brown clusters following (Rutherford and Xue, 2014). For the word embeddings, we use the mean of the vectors of all words.

In order to give an idea of the sparsity of the one-hot encodings, note that we have $|\mathcal{V}| = 33,649$ different tokens considering all implicit examples without filtering. The Brown clusters

[3] https://github.com/jkkummerfeld/nlp-util

[4] http://metaoptimize.com/projects/wordreprs/

[5] http://lebret.ch/words/

merge these tokens into $3,190$ codes (for $3,200$ clusters), 393 ($1,000$ clusters), 59 (320 clusters) or 16 (100 clusters). For heads, we count $5,615$ different tokens which correspond to $1,988$ codes for $3,200$ clusters and roughly the same number for the others. For the dense representations, the vocabulary size is twice the number of dimensions of the embedding, thus from 50 to 400, or the square of this number, thus from 625 to $40,000$.

Other Features We experiment with additional features commonly used for this task: productions rules, average verb phrases length, Levin verb classes, modality, polarity, General Inquirer tags, number, first last and first three words. These feature templates are well described in (Pitler et al., 2009; Park and Cardie, 2012). They all correspond to a one-hot encoding, except average verb phrases length which is continuous. We thus concatenate these features to the lexical ones.

4.2 Model

We use the same classification algorithm for comparing all the described feature configurations. Specifically, we train a Maximum Entropy (ME) classifier (aka, logistic regression).[6] As in previous studies, we build one binary classifier for each relation. In order to deal with class imbalance, we use a sample weighting scheme: each sample receives a weight inversely proportional to the frequency of its class in the training set. We optimize the hyper-parameters of the algorithm (i.e., the regularization norm: L_1 or L_2, and its strength) and a filter on the features on the development set, based on the F1 score. Note that filtering is pointless for purely dense representations. We test statistical significancy of the results using t-test and Wilcoxon test on a split of the test set in 20 folds.

Previous studies have tested several algorithms generally concluding that Naive Bayes (NB) gives the best performance (Pitler et al., 2009; Rutherford and Xue, 2014). We found that, when the hyper-parameters of ME are well tuned, the performance are comparable to NB if not better. Note that NB cannot be used with word embeddings representations as it does not handle negative value. Concerning the class imbalance issue, the downsampling scheme is the most spread since (Pitler et al., 2009) but it has been shown that oversampling and instance weighting lead to better performance (Li and Nenkova, 2014a).

[6] We use the implementation provided in Scikit-Learn (Pedregosa et al., 2011), available at: `http://scikit-learn.org/dev/index.html`.

Relation	Train	Dev	Test
Temporal	665	93	68
Contingency	3,281	628	276
Comparison	1,894	401	146
Expansion	6,792	1,253	556
Total	12,632	2,375	1,046

Table 2: Number of examples in train, dev, test.

4.3 Penn Discourse Treebank

We use the Penn Discourse Treebank (Prasad et al., 2008), a corpus annotated at the discourse level upon the Penn Treebank, giving access to a gold syntactic annotation, and composed of articles from the Wall Street Journal. Five types of examples are distinguished: implicit, explicit, alternative lexicalizations, entity relations, and no relation. Each example could carry multiple relations, up to four for implicit ones, and the relations are organized into a three-level hierarchy.

We keep only true implicit examples and only the first annotation. We focus on the top level relations which correspond to general categories included in most discursive frameworks. Finally, in order to make comparison easier, we choose the most spread split of the data, used in (Pitler et al., 2009; Park and Cardie, 2012; Rutherford and Xue, 2014) among others. The amount of data for training (sections $2-21$), development ($00, 01, 23, 24$) and evaluation ($21, 22$) is summarized in table 2.

5 Results

We first discuss the models that use only lexical features, defined either over all the words that appear in the arguments or only the head words. We then compare our best performing lexical configurations with the ones that also integrate additional standard features, and to state-of-the-art systems.

5.1 Word Pairs over the Arguments

Our first finding in this setting is that feature configurations that employ unsupervised word representations almost systematically outperform those that use raw tokens. This is shown in the left part of table 3. Although the optimal word representation differs from one relation to another, it is always a dense representation that achieves the

	All words				Head words only			
Representation	*Temp.*	*Cont.*	*Compa.*	*Exp.*	*Temp.*	*Cont.*	*Compa.*	*Exp.*
One-hot $\otimes$	21.14	50.36	34.80	59.43	11.96	43.24	17.30	**69.21**
One-hot $\oplus$	23.04	51.31	34.06	58.96	23.01	49.40	29.23	59.08
Brown $3,200$ $\otimes$	20.38	50.95	34.85	61.23	11.98	43.77	16.75	68.76
Best *Brown* $\otimes$	15.52	**53.85****	30.90	61.87	22.91	45.74	25.83	68.76
Best *Brown* $\oplus$	**27.96****	49.48	31.19	**67.42****	21.84	47.36	27.52	61.38
Best *Embed.* $\otimes$	22.97	52.76**	**34.99**	61.87	**23.88**	**51.29**	**30.59**	58.59
Best *Embed.* $\oplus$	25.98*	52.50	33.15	60.17	22.48	47.48	29.82	57.45

Table 3: F1 score for systems using all words and only heads for *Temporal* (*Temp.*), *Contingency* (*Cont.*), *Comparison* (*Compa.*) and *Expansion* (*Exp.*). * $p \leq 0.1$, ** $p \leq 0.05$ compared to *One-hot* $\otimes$ with t-test and Wilcoxon ; for head words, all the improvements observed against *One-hot* $\otimes$ are significant.

best F1 score. Our baselines correspond to multiplicative and additive one-hot encodings, noted *One-hot* $\otimes$ and *One-hot* $\oplus$, the former being the most commonly used in previous work. These are strong baselines in the sense they have been obtained after optimizing a frequency cut-off. Our best systems based on dense representations correspond to significant improvements in terms of F1 of about 8 points for *Expansion*, 7 points for *Temporal* and 3.5 for *Contingency*. The gains for *Comparison* are not statistically significant. All these results are obtained using the normalization to unit vectors possibly combined to the concatenation-specific normalization described in §3.3.

Comparing Dense Representations The best results are obtained using the Brown clusters (*Brown*) showing that this resource merges words in a way that is relevant to the task. Strikingly, the Brown configuration used in (Rutherford and Xue, 2014) (*One-hot Brown* $3,200$ $\otimes$) does not do better than the raw word pair baselines, except for *Expansion*. Recall that these authors did not explicitly provide this comparison. While doing a little worse, word embeddings (*Embed.*) also yield significant improvements for *Temporal* and *Contingency*, and smaller improvements for the others. This suggests that, even if they were not built based on discursive criteria, the latent dimensions encode word properties that are relevant to their rhetorical function. The superiority of Brown clusters over word embeddings is in line with the conclusions in (Turian et al., 2010) for two rather different NLP tasks (i.e., NER and chunking).

Turian et al. (2010) showed that the optimal word embedding is task dependent. Our experiments suggest that it is relation dependent: the best scores are obtained with *HLBL* for *Temporal*, *CnW* for *Contingency*, *H-PCA* for *Expansion* and *CnW* (Best *Embed.* $\otimes$) and *HPCA* (Best *Embed.* $\oplus$) for *Comparison*. This again demonstrates that these four relations have to be considered as four distinct tasks. Identifying temporal or causal links is indeed sensitive to very different factors, the former relying more on temporal expressions and temporal ordering of events whereas the latter relies on lexical and encyclopaedic knowledge on events. We think that this also explains that the behavior of the F1 against the optimal number of clusters for *Expansion* really differs from what we observed for the other relations: only 100 clusters for the best concatenated system and 320 for the best multiplicative one. *Expansion* is the least semantically marked relation and thus takes less advantage of fine-grained semantic groupings.

Comparing Word Combinations Comparing concatenated configurations ($\oplus$ systems) against multiplicative ones ($\otimes$ systems), we first note that for raw tokens the concatenated form (*one-hot* $\oplus$) yields results that are comparable, and sometimes better, than the standard multiplicative system (*one-hot* $\otimes$), while failing to explicitly model word pair interactions. With Brown clusters, the concatenated form *Best Brown* $\oplus$ lead to better F1 scores than *Best Brown* $\otimes$ except for *Contingency*.

When comparing performance on dev set, we found that the differences between concatenated and multiplicative forms for *Brown* (excluding *Expansion* for now) depend on the number of clusters used. Turian et al. (2010) found that the more clusters, the better the performance. This is also the case here with concatenated forms, but not with multiplicative forms. In that case, F1 increases un-

til $1,000$ clusters and then decreases. There is indeed a trade-off between expressivity and sparsity: having too few clusters means that we loose important distinctions, but having too many clusters leads to a loss in generalization. A similar trend is also found with word embeddings.

5.2 Head Words Only

Considering the right part of table 3, the first finding is that performance of systems that use only head words decrease compared to those using all words, but much more so with the baseline *One-hot* $\otimes$ than with other representations. *One-hot* $\otimes$ has very poor performance for most relations, losing between 7 and 17 points in F_1 score. The performance loss is much less striking with *One-hot* $\oplus$ and with denser representations, which are again the best performing. The only exception is *Expansion* whose precision however increases. As said, this relation is the less semantically marked, making it less likely to take advantage of the use of word representations. The best performance in this setting are obtained with word embeddings (not *Brown*) with significant gain from 8 to 13 points in F1 for most relations. Moreover, the best systems are all based on the multiplicative form confirming that this is a better way of representing pairs than simple concatenation when the number of initial dimensions is not too large.

5.3 Adding Other Features

Finally, we would like to assess how much improvement can still be obtained by adding other standard features, such as those in §4.1, to word-based features. Conversely, we want to evaluate how far we are from state-of-the-art performance by just using word representations. We compare our results with those presented in (Rutherford and Xue, 2014) and in (Ji and Eisenstein, 2014), both systems deal with sparsity either by using Brown clusters or by learning task-dependent representations. To make comparison easier we reproduce the experiments in (Rutherford and Xue, 2014) with Naive Bayes (NB)[7] and Maximum Entropy (ME) but without their coreference features and using gold syntactic parse. These correspond to the "repr." lines in table 4. We attribute the small differences observed with NB by the lack of coreference features and/or the use of different filtering thresholds. Concerning the difference between NB and ME, the only obvious issue is the low F1 score for *Expansion*: the system built using NB predicts all examples as positive thus leading to a high F1 score whereas the other one produces more balanced predictions, meaning neither systems is truly satisfactory. Finally, we give results using the traditional one-encoding based on word pairs plus additional features (*One-hot* $\otimes$ + additional features). These results are summarized in table 4, also including the best results of our experiments without additional features ("only").

Our first finding is that the addition of extra features to our previous word-based only configuration appears to outperform state-of-the art results for *Temporal* and *Contingency*, thus giving the best performance to date on these relations. These improvements are significant compared to our reproduced systems. Note that we also outperform the task-dependent embeddings of Ji and Eisenstein (2014), except for *Expansion*. Our tentative explanation for this is that these authors included Entity relations and coreference features. Note that our system corresponding to a reproduction of (Rutherford and Xue, 2014) gives results similar to the baseline using raw word pairs (*One-hot* $\otimes$ + additional features) showing that their improvements were due to other factors, the optimized filter threshold and the coreference features.

Overall, the addition of these hand-crafted features to our best systems do not provide improvements as high as one might have hoped. While improvements are significant compared to our reproduced systems, they are not with respect to the best systems given in table 3. When using all words, we only have a tendency toward significant improvement for *Contingency*[8]. These very small differences demonstrate that semantic and syntactic properties encoded in these features are already taken into account into the unsupervised word representations.

6 Related Work

Automatically identifying implicit relations is challenging due to the complex nature of the predictors. Previous studies have thus used many features relying on several external resources (Pitler et al., 2009; Park and Cardie, 2012; Biran and McKeown, 2013) as the MPQA lexicon (Wilson et al., 2005) or the General Inquirer lexicon (Stone and Hunt, 1963), or on constituent or dependency

[7] Implemented in Scikit-Learn, we optimized the hyper-parameter corresponding to the smoothing.

[8] $p = 0.135$ with ttest and $p = 0.061$ with Wilcoxon.

	Temporal	*Contingency*	*Comparison*	*Expansion*
System	F1	F1	F1	F1
(Ji and Eisenstein, 2014)	26.91	51.39	35.84	**79.91**
(Rutherford and Xue, 2014)	28.69	54.42	**39.70**	70.23
repr. (Rutherford and Xue, 2014) NB	28.05	52.95	37.38	70.23
repr. (Rutherford and Xue, 2014) ME	24.79	53.39	36.46	50.00
One-hot $\otimes$ all tokens + add. features	23.26	54.41	34.34	62.57
Best all tokens only	27.96	53.85	34.99	67.42
Best all tokens + add. features	**29.30**	**55.76**	36.36	61.76

Table 4: Systems using additional features ("+ add.features"), state-of-the art results either reported or reproduced ("repr.") using Naive Bayes ("NB") or logistic regression ("ME") and best system from previous table ("only").

parsers (Li and Nenkova, 2014b; Lin et al., 2009). Feature selection methods have been proved necessary to handle all of these features (Park and Cardie, 2012; Lin et al., 2009). Interestingly, Park and Cardie (2012) conclude on the worthlessness of word pair features, given the existence of such resources. We showed that provided unsupervised word representations, the opposite was in fact true, as dense word representations capture a lot of syntactic and semantic information.

The major problem of standard word pair representations is their sparsity. A line of work is to deal with this issue by adding automatically annotated data from explicit examples (Marcu and Echihabi, 2002), possibly using some kind of filtering or adaptation methods (Pitler et al., 2009; Biran and McKeown, 2013; Braud and Denis, 2014). Another line of work propose to make use of dense representations as Brown clusters in (Rutherford and Xue, 2014). These authors claim that this resource provides word representations that are relevant to the task, a conclusion that we considerably refined. Ji and Eisenstein (2014) propose to learn a distributed representation from the syntactic trees representing each argument in way that is more directly related to the task. Although this is an attractive idea, the score on top level PDTB relations are mostly below those reported by (Rutherford and Xue, 2014), possibly because their representations are learned on a rather small corpus, the PDTB itself, whereas building this kind of representation requires massive amount of data.

Our work also relates to studies comparing unsupervised representations for other NLP tasks such as name entity recognition, chunking (Turian et al., 2010), sentiment analysis (Lebret and Collobert, 2014) or POS tagging (Stratos and Collins, 2005). In particular, we found some similarities between our conclusions and those in (Turian et al., 2010). Our comparison is slightly richer in that it includes different methods of vector compositions and add an extra distributional representation to our comparison (namely, H-PCA).

7 Conclusions and Future Work

In this paper, we show that one can reach state-of-the-art results for implicit discourse relation identification using only shallow lexical features and existing unsupervised word representations thus contradicting previous conclusions on the worthlessness of these features. We carefully assess the usefulness of word representations for discourse by comparing various formulations and combination schemes, demonstrating the inadequacy of the previously proposed strategy based on Brown clusters and the distinctive relevance of head words, and by establishing that the created dense representations already provide most of the semantic and syntactic information relevant to the task thus alleviating the need for traditional external resources.

In future work, we first plan to extend our comparative framework to a larger set of relations and to other languages. We also want to explore methods for learning embeddings that are directly related to the task of discourse relation classification, potentially using existing embeddings as initialization (Labutov and Lipson, 2013). It is also clear that seeing discourse segments as bag of words is too simplistic, we would like to investigate ways of learning adequate segment-wide embeddings.

References

N. Asher and A. Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Chloé Braud and Pascal Denis. 2014. Combining natural and artificial examples to improve implicit discourse relation identification. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*.

Yangfeng Ji and Jacob Eisenstein. 2014. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Igor Labutov and Hod Lipson. 2013. Re-embedding words. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*.

Rémi Lebret and Ronan Collobert. 2014. Word emdeddings through hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Junyi Jessy Li and Ani Nenkova. 2014a. Addressing class imbalance for improved recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Junyi Jessy Li and Ani Nenkova. 2014b. Reducing sparsity improves the recognition of implicit discourse relations. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. 2009. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*.

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*.

Philip J. Stone and Earl B. Hunt. 1963. A computer approach to content analysis: Studies using the general inquirer system. In *Proceedings of the Spring Joint Computer Conference*.

Karl Stratos and Michael Collins. 2005. Simple semi-supervised pos tagging. In *Proceedings of NAACL Workshop on Vector Space Modeling for NLP*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*.