Expert Systems With Applications 165 (2021) 114130

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Extending latent semantic analysis to manage its syntactic blindness

Raja Muhammad Suleman [*], Ioannis Korkontzelos

Department of Computer Science, Edge Hill University, Ormskirk, Lancashire L39 4QP, United Kingdom

ARTICLE INFO

Keywords:
Natural Language Processing
Natural Language Understanding
Latent Semantic Analysis
Semantic Similarity

ABSTRACT

Natural Language Processing (NLP) is the sub-field of Artificial Intelligence that represents and analyses human language automatically. NLP has been employed in many applications, such as information retrieval, information processing and automated answer ranking. Semantic analysis focuses on understanding the meaning of text. Among other proposed approaches, Latent Semantic Analysis (LSA) is a widely used corpus-based approach that evaluates similarity of text based on the semantic relations among words. LSA has been applied successfully in diverse language systems for calculating the semantic similarity of texts. LSA ignores the structure of sentences, i.e., it suffers from a syntactic blindness problem. LSA fails to distinguish between sentences that contain semantically similar words but have opposite meanings. Disregarding sentence structure, LSA cannot differentiate between a sentence and a list of keywords. If the list and the sentence contain similar words, comparing them using LSA would lead to a high similarity score. In this paper, we propose xLSA, an extension of LSA that focuses on the syntactic structure of sentences to overcome the syntactic blindness problem of the original LSA approach. xLSA was tested on sentence pairs that contain similar words but have significantly different meaning. Our results showed that xLSA alleviates the syntactic blindness problem, providing more realistic semantic similarity scores.

1. Introduction

Natural Language Processing (NLP) is the sub-field of Artificial Intelligence that focusses on understanding and generating natural language by machines (Khurana et al., 2017). Formally, NLP is defined as “a theoretically motivated range of computational techniques for studying and representing naturally occurring texts (of any mode or type) at one or more levels of linguistic analysis for the purpose of attaining language that is like a human-like language processing for a range of tasks or applications” (Liddy, 2001). NLP is an interdisciplinary field lying at the intersection of computing science, computational linguistics, artificial intelligence and cognitive science. NLP is concerned with research and development of novel applications for Human Computer Interaction (HCI), with human languages as a medium of communication. NLP applications include human language understanding, lexical analysis, machine translation, text summarization, speech recognition, sentiment analysis, expert systems, question answering and reasoning, intelligent tutoring systems and conversational interfaces.

Calculating the similarity between text snippets is an important task for many NLP applications. Similarity scoring schemes range from basic string-based metrics to more complex techniques that employ semantic analysis. Simple string-based metrics only apply in cases of exact word matching. They do not consider inflection, synonyms and sentence structure. To capture these text variations, more sophisticated text processing techniques, able to calculate text similarity on the basis of semantics, are needed. Latent Semantic Analysis (LSA) is one such technique, allowing to compute the “semantic” overlap between text snippets. Introduced as an information retrieval technique for query matching, LSA performed as well as humans on simple tasks (Deerwester et al., 1990). LSA’s abilities to handle complex tasks, such as modelling human conceptual knowledge, cognitive phenomena and morphology induction have been assessed on a variety of tasks consistently achieving promising results (Landauer et al., 1998, 2007; Landauer & Dumais, 2008; Schone & Jurafsky, 2000). As its underlying principle, LSA considers the meaning of text in direct relationship with the occurrence of distinct words. Intuitively, LSA considers that words with similar meaning will occur in similar contexts. It has been used successfully in a diverse range of NLP applications (Landauer, 2002; Vrana et al., 2018; Wegba et al., 2018; Jirasatjanukul et al., 2019). For example, it has been extensively used as an approximation to human semantic knowledge and verbal intelligence in the context of Intelligent Tutoring Systems (ITS). LSA-based ITSs, such as AutoTutor and Write To Learn (Lenhard,

* Corresponding author.
E-mail addresses: sulemanr@edgehill.ac.uk (R.M. Suleman), yannis.korkontzelos@edgehill.ac.uk (I. Korkontzelos).

https://doi.org/10.1016/j.eswa.2020.114130
Received 30 January 2020; Received in revised form 26 July 2020; Accepted 14 October 2020
Available online 24 October 2020

2008), allow learners to interact with the system using a natural language interface. Even though LSA provides promising results in a multitude of applications, its major shortcomings come from the fact that it completely ignores syntactic information during similarity computations. LSA suffers the following inherent problems:

1) LSA is based on the semantic relations between words and ignores the syntactic composition of sentences. Consequently, it may consider semantically similar sentences with very different or even opposite meaning (Cutrone & Chang, 2011).
2) LSA does not consider the positions of subject and object of a verb as distinct, while comparing sentences. For example, LSA considers the sentences "The boy stepped on a spider" and "The spider stepped on a boy" as semantically identical, although they are semantically opposite to each other.
3) LSA considers list of words as complete sentences, despite the lack of proper structure (Islam & Hoque, 2010; Braun et al., 2017). For example, "boy spider stepped" is considered equivalent to the sentences in (2), and LSA considers them as semantically identical.
4) LSA does not consider negation. Consequently, it cannot differentiate between two semantically similar sentences, but one contains some negation. For example, "Christopher Columbus discovered America" and "Christopher Columbus did not discover America". Negation inverts the sentence's meaning. However, LSA assigns a similarity score of more than 90% to this pair of sentences.

In this paper, we explore ways to enrich LSA with syntactic information to enhance its accuracy when comparing short text snippets. We employ Parts-Of-Speech (PoS) tags and Sentence Dependency Structure (SDS) to enrich the input text with syntactic information. Current trends in NLP research focus on Deep Learning. Neural network-based architectures are employed to model complex human behaviors in natural language. These methods have achieved top performance levels for many semantic understanding tasks, arguably due to their ability to capture syntactic representations of text (Gulordava et al., 2018; Kuncoro et al., 2018; Linzen, Emmanuel, & Yoav, 2016; Hewitt and Manning, 2019). Lately, a variety of models that produce embeddings that capture the linguistic context of words have been proposed, ranging from Word2vec (Mikolov, 2013) to state-of-the-art transformer-based architectures, such as BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019). We evaluate our method against some of the current neural network-based methods, such as Universal Sentence Encoder (USE) (Cer, 2018), Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) and XLNet (Yang et al., 2019). The results show that xLSA performs consistently better than these techniques on short text snippets.

The rest of the paper is organized as follows: Section 2 provides an overview of text similarity approaches and describes research work on enriching the LSA model with syntactic information. Section 3 introduces xLSA and provides details about the proposed extension to LSA. Section 4 describes the experimental settings and summarizes the results of the comparative analysis. Section 5 concludes the findings and proposes directions for future work.

2. Background and related work

NLP methods aim at allowing computers to understand and manipulate language like humans do. NLP applications have been successful in opening new dimensions of Human-Computer Interactions (HCI).

2.1. Natural language understanding

Natural Language Understanding (NLU) deals with tasks that extract structured semantic information from unstructured text or speech (Braun et al., 2017). NLU breaks down natural language into a structured ontology, allowing computers to understand it and identify artefacts such as intents, semantics, sentiments etc. In the last years, NLU is an active area of research, due to its applications to HCI, especially with the recent popularity of semantic search and conversational interfaces, also known as chatbots (Pereira & Díaz, 2019).

2.2. Text similarity approaches

String similarity can be measured based on lexical or semantic analysis. Strings are lexically similar if they consist of the same sequence of characters. They are semantically similar if they have same meaning or are used in similar contexts. String-based similarity is evaluated on character composition and word sequence, whereas corpus-based similarity is evaluated on the basis of a large corpus. Knowledge-based similarity is determined on the basis of information in a semantic network (Gomaa & Fahmy, 2013).

2.3. String-based similarity

String-based similarity measures can be split in two major categories: character-based similarity measures and term-based similarity measures. Longest Common SubString (LCS) and Damerau-Levenshtein are among the most popular string-based similarity techniques. Character-based techniques are of limited applicability, because they can only capture exact matches. Cosine and Jaccard similarity are two commonly used term-based similarity techniques.

2.4. Corpus-based similarity

Corpus-based similarity approaches compute semantic similarity between two strings or words based on information gathered from a corpus, i.e., a large collection of text. Hyperspace Analogue to Language (HAL), Pointwise Mutual Information – Information Retrieval (PMI-IR) and Latent Semantic Analysis are some of the most popular corpus-based similarity approaches.

2.5. Knowledge-based similarity

Knowledge-based similarity approaches evaluate word similarity based on information retrieved from semantic networks. WordNet is the most popular sematic network for computing knowledge-based similarity. Knowledge-based similarity measures can be divided in two categories: semantic similarity measures and semantic relatedness measures. Two words are semantically similar if they have the same meaning or are synonymous. Two words are semantically related if they are used in proximity. Mohler and Mihalcea (2009) provided a comparison between corpus-based and knowledge-based similarity measures. According to their findings, the corpus-based approaches can be improved by accounting for corpus size and domain.

2.6. Latent semantic analysis (LSA)

LSA considers the meaning of a document or a passage as directly associated to the occurrence of particular words in it. Kuechler (2007) provided a detailed overview of a number of information systems and business applications of textual data analysis that use LSA. It has been extensively used in reviewing the literature quantitatively, in computer-mediated textual data analysis, in customer feedback and interview analysis, and in knowledge repositories management (Evangelopoulos et al., 2012). LSA assumes that words that have similar meaning are likely to occur in related pieces of text. LSA starts by populating a matrix of word count per sentence or paragraph. Each column represents a sentence or paragraph and each row represents a unique word. Singular Value Decomposition (SVD), a well-known dimensionality reduction method, is used to reduce the number of columns, preserving the similarity structure among rows. Words are matched by calculating the cosine similarity between two vectors, which ranges between zero and one (Landauer & Dumais, 1997). LSA-based automated text grading systems have been shown to outperform or at least perform comparably with human graders in multiple experiments (Toutanova et al., 2003).

A range of approaches have been applied to enhance LSA with morphological and syntactic knowledge. The Tagged LSA (TLSA) (Wiemer-Hastings et al., 2001) added synthetic information to LSA. It considered a word together with its PoS tag as a single term, whereas the original LSA does not differentiate between different part of speech of the same word. The Syntactically Enhanced LSA (SELSA) (Kanejiya et al., 2003) is similar to the TLSA. This method populates a matrix where each row consists of a focus word and the PoS of the previous word and each column corresponds to a document or sentence. The Parts-Of-Speech Enhanced LSA (POSELSA) (Kakkonen et al., 2006) focused on enhancing LSA by adding PoS information. The technique used three Word by Context Matrices (WCM) for each word. The first entry was for the PoS tag of a focus word, the second entry was for the PoS tags of the focus word and its preceding word, whereas the third entry was for the PoS tag of the focus word and its succeeding word. Results showed that using parts-of-speech improved accuracy by 5% to 10% in comparison to the original LSA. However, the computational complexity of POSELSA was very high. The Polarity Inducing LSA (Yih et al., 2012) introduced the notion of polarity, allowing the system to handle two opposite relationships between words, i.e., synonyms and antonyms. As part of a modified LSA that was applied for automatic Arabic essay scoring, TF-POS was proposed. TF-POS is a transformed version of Term Frequency-Inverse Document Frequency (TF-IDF) that combines PoS tagging with TF to add syntactic information into the vectors of words (Mezher & Omar, 2016). The model was trained on 488 student answers and tested on 183 answers. The results showed enhancements in the Modified LSA score, when compared to original LSA scores.

The methods for enhancing LSA, that were described above, add syntactic information to the data used to train LSA models. Some approaches have only used PoS tags, whereas others have combined PoS tags with SDS information to enrich their training data. Our research focuses on the task of calculating semantic similarity of short sentences. To address it, we introduce a wrapper around LSA. We do not train our own model, but use an existing LSA model trained on the UMBC WebBase corpus (Han et al., 2013). We use syntactic information of the input tokens to generate corresponding candidates for LSA comparison. The results of token-pair comparisons are then combined to generate an overall semantic similarity score for the input sentences.

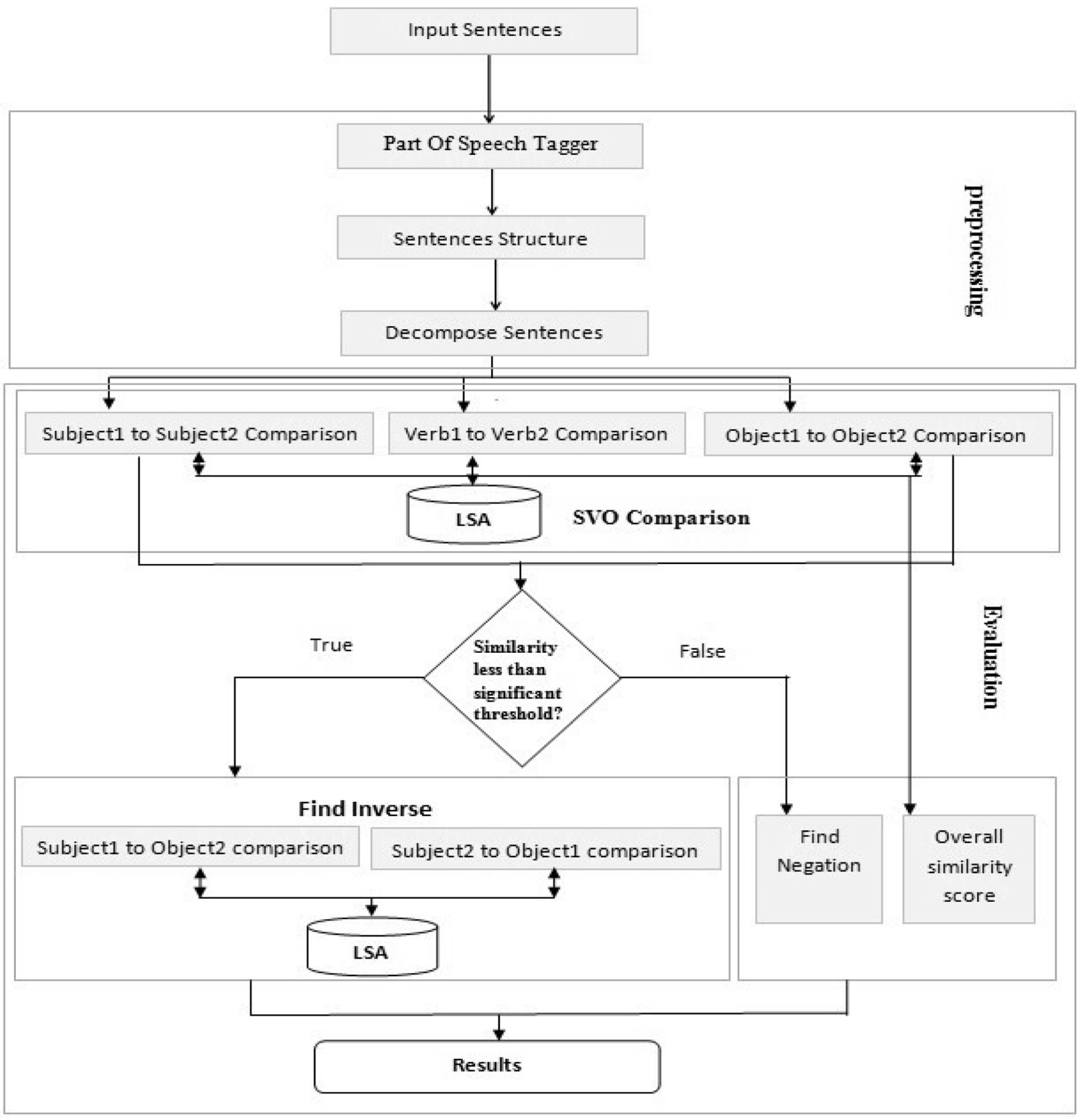
Input Sentences
Part Of Speech Tagger
Sentences Structure
Decompose Sentences
preprocessing
Subject1 to Subject2 Comparison
Verb1 to Verb2 Comparison
Object1 to Object2 Comparison
LSA
SVO Comparison
True
Similarity less than significant threshold?
False
Evaluation
Find Inverse
Subject1 to Object2 comparison
Subject2 to Object1 comparison
LSA
Find Negation
Overall similarity score
Results

Fig. 1. xLSA Execution Flow.

3. Extended Latent semantic analysis (xLSA)

3.1. xLSA overview

Given LSA's syntactic blindness problem, we propose an algorithmic extension to address it by combining Sentence Dependency Structure (SDS) and Parts-of-Speech (PoS) tags. The proposed algorithm has been developed for the English language and validated over a test set of sentences, collected from various corpora (Bowman et al., 2015; Young et al., 2014). Fig. 1 shows the flow of the proposed system.

A sentence is a complete thought in written, consisting of a subject and a predicate. The subject is "a person, thing or place that is performing some action", whereas the predicate describes this action. The simplest form of the predicate is just a verb, e.g., in the sentence *"I breathe."*. A more complex predicate can include an object, i.e., "a noun or pronoun that can be affected by the action of a subject". Simple sentences in English follow the Subject-Verb-Object (SVO) rule, where the verb shows the relationship between the subject and the object. xLSA uses SDS and PoS tags to identify the Subject, Verb and Object in a sentence along with their asymmetric relationships. This information is used to calculate the similarity of two sentences, by matching the SVO structure. xLSA works in two phases: (i) the pre-processing phase and (ii) the evaluation phase. The pre-processing phase tokenises the input sentences and assigns a PoS tag to each token. For each input sentence, it also computes its SDS, that is then used in the evaluation phase to determine the structural similarity among the input sentences.

Our method uses PoS tagging, SDS and Sentence Decomposition to enrich the sentences for comparison. For tokenisation and PoS tagging, the system uses the spaCy Tokenizer and the spaCy PoS tagger, respectively (Honnibal & Johnson, 2015).

PoS ambiguity refers to cases where the same form of a word may occur in text with different PoS. For example, in the sentence "the boy steps on the spider", "steps" is a verb describing the boy's action, whereas in the phrase "the steps are broken" the same word is a noun. PoS ambiguous words, such as "steps", can have different PoS and meaning dependent on their context. The spaCy PoS tagger can handle PoS ambiguity, and Fig. 2 shows an example.

Dependency grammar is a class of modern syntactic theories, based on dependency relation (Nivre, 2005). Dependency is a formalism that represents relations between words as directed links. Verbs are considered structural centres of clauses and other structural units link with verbs by directed links. Dependency structure provides information about grammatical functions of words in relation to other words in the sentence. The English language has four types of sentences:

1) Simple sentences
2) Compound sentences
3) Complex sentences
4) Compound-complex sentences

Complex, compound and compound-complex sentences are broken down into simple sentences for SVO comparison (Adhya & Setua, 2016). The spaCy library is used to generate the Dependency Structure of input sentences. Dependency Structure provides the system with information about sentence structure, to ensure that the provided input has a proper sentence format. The results are used to check whether the input is a proper sentence or an arbitrary list of words.

During Decomposition, the sentences are split into subjects, verbs and objects. In case of active-voice sentences, the spaCy library uses the "Nominal Subject" and "Direct Object" tags to specify the subject and object, respectively. To deal with the passive-voice sentences, spaCy denotes subjects and objects as "Nominal Subject (Passive)" and "Object of Preposition", respectively. As mentioned earlier, the spaCy library is capable of resolving PoS ambiguity, ensuring that the *root* verb of a sentence is identified correctly. Nouns that are the right descendants of the root verb are considered as objects and nouns that appear before the root verb, i.e., the left descendants, are considered as subjects. At this stage, we compute Subject-Verb Agreement (SVA) for each sentence. SVA is an NLP task useful for Grammatical Error Detection (GED) (Leacock et al., 2010; Enguehard et al., 2017; Wang and Zhao, 2015). SVA entails that the subjects and verbs in a sentence must agree on their multiplicity, i.e., a singular verb takes a singular subject and a plural verb takes a plural subject. During sentence decomposition, we create a list of subjects and verbs in the input sentences along with their relational dependencies. This information is used to assign an SVA flag to each sentence, specifying whether there is number agreement between its subject and verb.

In succession, we check input sentences for negation. To denote this the spaCy Dependency Parser assigns a negation relationship ("neg") between the auxiliary (AUX) token and the particle (PART) token. We use this information to check whether both sentences are negated or if only one of them is. In the latter case, we update the isNegated flag to highlight the negation disagreement.

3.2. Evaluation

SVO Comparison: After decomposition, sentences are compared on the basis of subject, verb and object (Ab Aziz, et al., 2009; Adhya & Setua, 2016; Wiemer-Hastings et al., 2001). Before the comparison, the list of subjects, verbs and objects are stemmed. Stemming maps words to their base form and allows easy comparison between different inflections of a word (Cutrone & Chang, 2011). For example, the common base form of "processing" and "processed" is "process". Stemming is applied to simplify the process of matching terms. After stemming, xLSA performs a cross-comparison, i.e., compares subject(s) of the first sentence with the subject(s) of second sentence, the verb(s) of the first sentence with the verb(s) of the second sentence and the object(s) of first sentence with the object(s) of the second sentence. If the first sentence has an object and the second sentence does not, the similarity score of objects is set to zero. To compute similarity of subjects, verbs and objects, it is necessary that they exist in both sentences. If they only exist in one sentence, then the similarity score is set to zero.

To compute similarity, we used the UMBC STS (Semantic Textual Similarity) Service API.[1] UMBC STS uses a hybrid approach, combining distributional similarity and LSA to compute word similarity. The UMBC service was evaluated on different token-pairs to determine the upper and lower bounds of acceptance thresholds ranging from 0.1 to 1.0, with a 0.1 increment between consecutive tests. Similarity scores of less than 0.4 (40%) were observed for tokens that were completely unrelated with no semantic relevance, whereas similarity scores of greater than 0.7 (70%) were consistently observed for tokens that were semantically or contextually similar. If subject-to-subject and object-to-object similarity scores for the two sentences are less than the minimum threshold, then xLSA cross-compares the subjects and objects. If the cross-similarity scores for the subjects, objects and verbs for both sentences is greater than or equal to the upper threshold value then xLSA sets the inverse flag to '1' for the pair of sentences.

After computing the inverse flag, xLSA similarity for complete sentences is calculated as the average method (Wiemer-Hastings et al., 2001). xLSA similarity provides a measure of semantic and syntactic similarity of the sentences. The score is averaged with respect to the number of subjects, objects and verbs of the sentences. For sentences that are found to be semantically similar based on SVO comparison, the isNegated flag specifies whether one of the sentences in the pair negates the other.

[1] UMBC STS Service is available at: swoogle.umbc.edu/SimService.

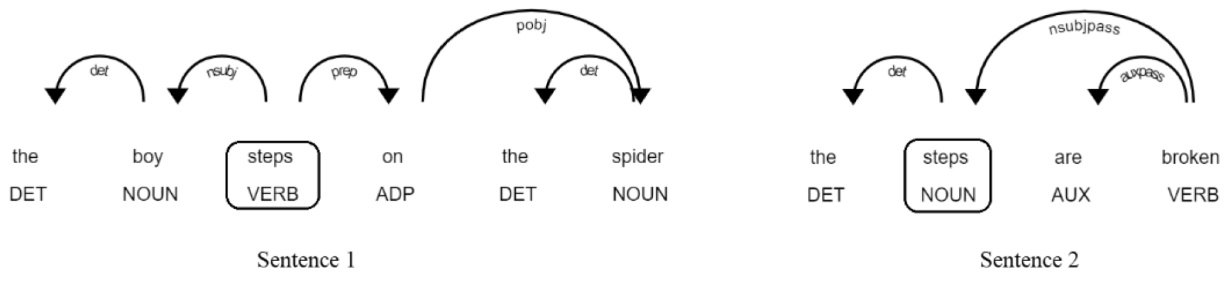

Fig. 2. The spaCy POS tagger produces different SDSs for the PoS ambiguous word "steps".

4. Experiments and results

4.1. Dataset

For our experiments, we used sentences from two publicly available datasets (Bowman et al., 2015; Young et al., 2014). The datasets contain pairs of English sentences that are semantically similar to each other. The SNLI corpus (Bowman et al., 2015), is a collection of 570 k human written English sentences. The Flickr corpus (Young et al., 2014), contains 30 k English sentences. The selection criterion for sentence pairs was semantic relatedness, for both corpora.

We selected three categories of sentences. The first category contains semantically similar pairs of sentences. The second category contains pairs of sentences with similar words, but complete opposite (inverse) meaning. The third category contains pairs of semantically related sentences, where one of the sentences had a negation in it. All three categories include sentences in active voice or passive voice, and can contain multiple subjects, verbs and objects. Some sentences have no verb, instead of which only helping verb was used in the sentence. A few sentences also included 'gerunds' which can be used as a noun or a verb depending upon the context of the sentence.

5. Experiments

5.1. Experiment I

The first experiment compared pairs of semantically similar sentences, where one sentence was in active voice and the second sentence was in passive voice. We considered pair of sentences such as "the boy is stepping on a spider" and "the spider is being stepped on by a boy". Table 1 shows the PoS tags for each sentence.

After PoS tagging, xLSA decomposed the sentences into subjects, verbs and objects on the basis of their dependency structures. Table 2 shows the generated SVO structure. The subject, verb and object of the first sentence were compared with the subject, verb and object of the second sentence. This comparison was used to calculate the similarity scores between the two sentences on the basis of SVO values. Table 3 shows the Subject Similarity Score (SubSim Score), Objects Similarity Score (ObjSim Score) and Verbs Similarity Score (VerbSim Score). The xLSA Similarity Score (xLSA Score) was calculated by using an averaging formula. In this scenario, xLSA and LSA had equal sentence similarity scores, since the sentences had the same meaning. Inverse and Negation flags were set to zero, since the sentences have the same subjects and objects and none of them is negated.

Table 1
Experiment I: Words with corresponding PoS tags.

First Sentence								
The	boy	is	stepping	on	a	Spider		
DT	NN	VBZ	VBG	IN	DT	NN		
Second Sentence								
The	spider	is	being	stepped	on	by	a	boy
DT	NN	VBZ	VBG	VBN	IN	IN	DT	NN

Table 2
Experiment I: SVO structure of sentences.

Sentences	Subject	Verb	Object
Sentence-1	Boy	Stepping	Spider
Sentence-2	Boy	Stepped	Spider

Table 3
Experiment I: LSA vs xLSA.

SubSim Score	VerbSim Score	ObjSim Score	xLSA Score	LSA Score	Inverse	Negation
1	1	1	1	1	0	0

5.2. Experiment II

In second experiment, we compared semantically related sentences with inverse meaning. For example, let us consider the sentences: "The cat climbs on the tree" and "The tree climbs on the cat". Table 4 shows the PoS tags of their words.

After PoS tagging, the dependency structure of the sentences were generated. Then, the sentences were decomposed into subjects, verbs and objects on the basis of their dependency structure, as shown in Table 5. The Subject, Verb and Object similarity scores were computed by matching "subject to subject", "verb to verb" and "object to object". If the similarity score of subjects and objects is less than 40% and the verb similarity score is greater than 70%, then a cross-comparison of the subject of the first sentence with object of the second sentence and the object of the second sentence with the subject of the first sentence is performed. If the cross-similarity score is greater than 70%, then the inverse flag is set to one. The default value for the inverse flag is zero. Table 6 shows that LSA computed a similarity score of 100%, whereas xLSA assigned a similarity score of 44% to this sentence pair. In addition, xLSA detected that the two sentences were inverse of each other and set the Inverse flag to one.

5.3. Experiment III

The third experiment dealt with semantically similar sentences, where each sentence negates the other. For example, we considered sentences such as: "Alessandro Volta invented the battery" and "Battery was not invented by Alessandro Volta". Table 7 shows the words and PoS tags of the sentences. Then, the sentence dependency structures were

Table 4
Experiment II: Words with corresponding PoS tags.

First Sentence					
The	cat	climbs	on	the	tree
DT	NN	VBZ	IN	DT	NN
Second Sentence					
The	tree	climbs	on	the	cat
DT	NN	VBZ	IN	DT	NN

Table 5
Experiment II: SVO structure of sentences.

Sentences	Subject	Verb	Object
Sentence-1	cat	climbs	tree
Sentence-2	tree	climbs	cat

Table 6
Experiment II: LSA vs xLSA Scores.

SubSim Score	VerbSim Score	ObjSim Score	xLSA Score	LSA Score	Inverse	Negation
0.16	1	0.16	0.44	1	1	0

used to determine voice as well as the subjects, verbs and objects, as shown in Table 8.

The subjects, verbs and objects were stemmed to their base forms that were then compared to compute semantic relatedness. The xLSA similarity score was evaluated based on the similarity between subjects, verbs and objects. If the xLSA similarity score is greater than 0.7, then xLSA checks negation in both sentences. If one of the sentences is negated, then the negation flag is set to one otherwise it remains zero. Table 9 shows that LSA computed a similarity score of 83% for this pair of sentences. xLSA computed a similarity score of 100% since it was able to handle the change in the voice of the sentences. The LSA score was less than the xLSA score because it took the adverb "not" into account during the computation. LSA was unable to identify that the sentences were semantically related and that the second sentence was negated, which inverted its meaning. xLSA produced a similarity score of 100% because it only considered the subject, verb and object in each sentence and set the negation flag to one. This means that xLSA identified that the sentences are semantically similar but mutually contradicting, as one of them is negated.

5.4. *Experiment IV*

In the fourth experiment we compared sentences with similar words, where one of the words appears with a different PoS in each sentence. For example, consider the sentences "john writes a report" and "john reports a murder". In the first sentence, "report" is a noun, whereas in the second sentence "reports" is a verb. Table 10 shows the words and PoS tags of the sentences. In the first sentence, "writes" was tagged as the verb and "report" was tagged as a noun, which qualifies it as an object. In the second sentence, "reports" was tagged as a noun and no verbs were found. For sentences without verbs, xLSA counts the number of nouns and if it is greater than one then it matches the list of nouns (subjects) with a pre-defined array of verbs whose forms are also used as nouns. If a match is found, xLSA considers that noun as a verb and marks its position. In the second sentence, the word "reports" was marked as the verb. In succession, the subjects, verbs and objects in the sentences were identified as shown in Table 11. The stemmed subjects, verbs and objects were then compared to compute similarity scores, as shown in Table 12. The two sentences were quite different from each other on the semantic level, however LSA assigned a similarity score of 72%. xLSA gave a similarity score of 52%, which is below the acceptable similarity threshold of 70% for our scheme.

5.5. *Experiment V*

As mentioned in the beginning of this paper, current NLP research has mainly focussed on Deep Learning methods, which exploit neural networks to learn representations of text in order to solve NLP tasks. Many state-of-the-art methods have shown promising results on a variety of NLP tasks such as Text Classification, Name Entity Recognition, Semantic Role Labelling, Grammatical Error Detection, Information extraction, Intent Detection and Slot Filling, Language modelling etc. A survey on the applications of Deep Learning for NLP provides an insight into the depth and breadth of current NLP research (Otter et al., 2020). Since the focus of our study is LSA, i.e., a statistical approach, it makes sense to see how it would compare to the more recent techniques. We evaluated our approach against some of the current well-known publicly available NLP models: Google's USE, BERT and XLNet on the task of computing semantic similarity for short/simple English sentences. USE adopts a transformer-based architecture, able to handle context in text spans. This allows USE to generate sentence-level embeddings. BERT is also based on a transformer architecture that uses an attention mechanism to learn contextual relations amongst tokens in text. BERT uses encoders and decoders to read text and generate predictions,

Table 8
Experiment III: SVO structure of sentences.

Sentences	Subject	Verb	Object
Sentence-1	alessandro volta	invented	battery
Sentence-2	alessandro volta	invented	battery

Table 9
Experiment III: LSA vs xLSA Scores.

SubSim Score	VerbSim Score	ObjSim Score	xLSA Score	LSA Score	Inverse	Negation
1	1	1	1	0.83	0	1

Table 10
Experiment IV: Words with corresponding PoS tags.

First Sentence			
John	Writes	a	report
NNP	VBZ	DT	NN
Second Sentence			
John	Reports	a	murder
NNP	NNS	RB	NN

Table 11
Experiment IV: SVO structure of sentences.

Sentences	Subject	Verb	Object
Sentence-1	john	writes	report
Sentence-2	john	reports	murder

Table 7
Experiment III: Words with corresponding PoS tags.

First Sentence						
Alessandro	volta	invented	the	battery		
NNP	NNP	VBD	DT	NN		
Second Sentence						
Battery	was	not	invented	by	alessandro	volta
NN	VBD	RB	VBN	IN	NNP	NNP

Table 12
Experiment IV: LSA vs xLSA Scores.

SubSim Score	VerbSim Score	ObjSim Score	xLSA Score	LSA Score	Inverse	Negation
1	0.47	0.11	0.52	0.72	0	0

respectively. XLNet is a generalized autoregressive pre-training method that exploits directional dependencies of context words to predict the following words in text. It also makes use of a transformer architecture, in particular Transformer XL (Dai et al., 2019), to learn long-term dependencies.

For this evaluation, we used the same sentence-pairs that were used in the evaluation of xLSA against simple LSA. Google provides pre-trained models for USE along with an interactive Jupyter Notebook on their cloud-based environment, Google Colab.[2] The notebook code was not modified, apart from adding a command to display the similarity results as real numbers rather than plotting them on a heatmap. Similarly, spaCy also provides a Jupyter notebook for the Colab environment that allows to use spaCy's implementation of BERT and XLNet models.[3] Again, the models were used out-of-the-box, i.e., no code changes or parameter tuning was performed for any of the models. Table 13 shows the results of a single sentence-pair comparison, while the complete evaluation is presented in the next section. xLSA assigns the lowest and most accurate similarity score to this sentence pair.

6. Results

The experiments discussed in section 4.2 highlight xLSA's capability to handle frequently occurring scenarios in text matching. Due to subtle ambiguities in natural language, the results of semantic similarity measures can be unpredictable. Sentences may be assessed as highly similar due to the occurrence of common terms, but still have completely different meaning. For evaluation, we tested xLSA against simple LSA, Google's USE, BERT and XLNet on a set of 100 sentence-pairs. Table 14 provides the average similarity scores produced by each technique.

6.1. xLSA vs LSA

Simple LSA gives a semantic similarity score of 100% to all the sentences that have similar words, irrespectively of the effect they have on the meaning of a sentence. xLSA has been designed to calculate semantic similarity not only based on similar words, but also on the syntactic structure of the sentences and the positioning of words in them. This allows xLSA to distinguish between sentences that are semantically related on the surface level, i.e., based on the words that they contain, but convey completely different meaning.

LSA does not consider the impact of negation on the meaning of sentences, therefore it fails to identify similarity correctly, when one of the sentences is negated. Using xLSA, all sentence pairs in the test set that contained at least one negation sentence were identified successfully. This means that two sentences might have a high semantic relatedness score, since they have common words, however, if one of the sentences negates the other, then the semantic similarity between them is adjusted to address this negation. Table 15 shows some examples of inverse sentences in the test set, which were successfully flagged as inverse by our algorithm.

LSA also does not consider the syntactic structure of sentences during comparison. This means that comparing a complete sentence with a list of words can yield a similarity score as high as 100%. This might be counter-intuitive for applications that require proper sentences to be matched, e.g., automated answer grading systems. To overcome this, xLSA not only tests the sentences on the semantic level, but proper syntactic structure is also validated to ensure that the input is not a list of keywords. xLSA has identified all such instances successfully.

6.2. xLSA vs Deep learning-based techniques

We evaluated xLSA against 3 Deep Learning-based (DL) models: USE, BERT and XLNet on the same set of sentence-pairs that was used to evaluate xLSA against simple LSA. The dataset contained short simple sentences in English. xLSA along with all the DL models provided high similarity scores for sentences that were semantically similar. DL models use contextual word embeddings to analyse the meaning of text and compute similarity. Following simple LSA, these approaches overlook changes in the sentence structure. For example, the sentence-pair "the cat climbed a tree" and "a tree climbed the cat" have complete opposite meanings, however all of the DL models gave a greater than 85% similarity score to this pair, and also all other similar sentence pairs in the test set. In addition, these models do not capture negation, hence sentences such as "the cat climbed the tree" and "the cat did not climb the tree" receive a greater than 85% similarity score. On the other hand, since DL models are trained on huge amounts of textual data, they able to generalize better and perform well for sentences with ambiguous structures.

7. Conclusion

Natural language carries huge complexity and uncertainty due to its ambiguous nature. This makes automated analysis and extraction of useful information from a given text a very difficult task. Natural Language Understanding (NLU) has garnered a lot of research interest due to its use in achieving seamless virtual conversational interfaces. Understanding text forms the basis of many advanced NLP tasks, and requires systems to gracefully manage the ambiguities of natural language. Latent Semantic Analysis (LSA) is a corpus-based approach that computes similarity of text within a corpus using algebraic techniques. LSA is used in document classification, semantic search engines, automated short answers grading and many more tasks. LSA-based evaluation has been shown to correlate strongly with human grading results (Gutierrez et al., 2013). LSA considers the semantic relationship among words, but it overlooks the structure of a sentence, which may cause a logically wrong answer to be treated as correct. Syntax plays a key role in understanding the meaning of a sentence and traditional LSA is blind to it.

To mitigate LSA's syntactic blindness problem, this paper aimed to provide an extension to LSA (xLSA), focussing on syntactic composition as well as the semantic relations in sentences. xLSA analyses sentences to identify their proper sentence structure using Sentence Dependency Structures (SDS) and the positioning of Parts-of-Speech (PoS) tags. If the sentences have a proper structure, then xLSA focuses on dependency structures and decomposes each sentence into Subject, Verb and Object (SVO). The sentences are compared based on the similarity between the SVOs. xLSA can identify inverse sentences by cross comparing subjects and objects of the two sentences. xLSA also identifies negation in a pair of semantically related sentences, where one of the sentence negates the other.

In English, many words are PoS ambiguous, i.e., can be used both as verbs and nouns. Most PoS taggers cannot differentiate among these words in a sentence. xLSA addresses this problem during the dependency structure phase, by using a list of words that can be used both as nouns and verbs. Our solution is limited to this list of PoS ambiguous words. We have tested xLSA with semantically similar sentences from two corpora against simple LSA and 3 Deep Learning models. xLSA's results are very promising, but are limited by the number and categories of

[2] colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/semantic_similarity_with_tf_hub_universal_encoder.ipynb.

[3] colab.research.google.com/github/explosion/spacy-pytorch-transformers/blob/master/examples/Spacy_Transformers_Demo.ipynb.

Table 13
Experiment V: xLSA vs USE, BERT, XLNet.

		USEScore	BERTScore	XLNetScore	xLSAScore
Sentence-1Sentence-2	John writes a reportJohn reports a murder	0.63	0.78	0.78	**0.52**

Table 14
Evaluation Results.

Sentence-Pair Type	USEScore (Avg.)	BERTScore (Avg.)	XLNetScore (Avg.)	LSAScore (Avg.)	xLSAScore (Avg.)
Similar	0.90	0.92	0.98	**0.99**	0.96
Inverse	0.88	0.87	0.97	1.0	**0.38**
***Negated**	**	**	**	**	**1.0**$^+$

*Checked only when sentences have a high similarity score**Negation not handled$^+$xLSA captured all negative sentences correctly

Table 15
Inverse sentences similarity scores.

Sentence Pair	LSA Score	xLSA Score	Inverse
the earth must revolve around the sun.the sun must revolve around the earth.	1	0.55	1
koko was asked to choose a house or a tree.a house or a tree were asked to choose koko.	1	0.34	1
money cannot buy happiness.happiness cannot buy money.	1	0.36	1
the hard disk stores data.the data stores hard disk.	1	0.42	1
the cat climbs on a tree.the tree climbs on a cat	1	0.44	1
the dog bit a child.the child bit a dog.	1	0.47	1
tom is writing a letter and a book.letter and book are writing tom.	1	0.33	1

sentences in the test set that was used for evaluation. We aim to address these limitations in the future, by increasing the types of sentences that can be handled by xLSA and running a more thorough evaluation.

CRediT authorship contribution statement

Raja Muhammad Suleman: Conceptualization, Methodology, Investigation, Software. **Ioannis Korkontzelos:** Supervision, Methodology, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research work is part of the TYPHON Project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 780251.

References

Ab Aziz, M. J., Dato'Ahmad, F., Ghani, A. A. A. \& Mahmod, R. (2009, October). Automated marking system for short answer examination (AMS-SAE). In 2009 IEEE symposium on industrial electronics & applications (Vol. 1, pp. 47–51). IEEE.

Adhya, S. \& Setua, S. K. (2016). Automated short answer grader using friendship graphs. In Computer science and information technology-proceedings of the sixth international conference on advances in computing and information technology (ACITY 2016) (Vol. 6, No. 9, pp. 13–22).

Bowman, S. R., Angeli, G., Potts, C. \& Manning, C. D. (2015). A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

Braun, D., Mendez, A. H., Matthes, F. \& Langen, M. (2017, August). Evaluating natural language understanding services for conversational question answering systems. In Proceedings of the 18th annual SIGdial meeting on discourse and dialogue (pp. 174–185).

Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., John, R. S. \& Sung, Y. H. (2018). Universal sentence encoder. arXiv preprint arXiv:1803.11175.

Cutrone, L. \& Chang, M. (2011, July). Auto-assessor: computerized assessment system for marking student's short-answers automatically. In 2011 IEEE international conference on technology for education (pp. 81–88). IEEE.

Deerwester, Scott, Dumais, Susan T., Furnas, George W., Landauer, Thomas K., & Harshman, Richard (1990). Indexing by latent semantic analysis. *Journal of the American society for information science, 41*(6), 391–407.

Devlin, J., Chang, M. W., Lee, K. \& Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V. \& Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.

Enguehard, E., Goldberg, Y. \& Linzen, T. (2017). Exploring the syntactic abilities of RNNs with multi-task learning. arXiv preprint arXiv:1706.03542.

Evangelopoulos, N., Zhang, X., & Prybutok, V. R. (2012). Latent semantic analysis: five methodological recommendations. *European Journal of Information Systems, 21*(1), 70–86.

Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications, 68*(13), 13–18.

Gutierrez, F., Dou, D., Martini, A., Fickas, S. \& Zong, H. (2013, December). Hybrid ontology-based information extraction for automated text grading. In 2013 12th International conference on machine learning and applications (Vol. 1, pp. 359–364). IEEE.

Han, L., Kashyap, A. L., Finin, T., Mayfield, J. \& Weese, J. (2013, June). UMBC_ EBIQUITY-CORE: Semantic textual similarity systems. In Second joint conference on lexical and computational semantics (*SEM), Volume 1: Proceedings of the main conference and the shared task: Semantic textual similarity (pp. 44–52).

Gulordava, Kristina, et al. (2018). Colorless green recurrent networks dream hierarchically. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1195–1205. In press.

Hewitt, J. \& Manning, C. D. (2019, June). A structural probe for finding syntax in word representations. In Proceedings of the 2019 conference of the North American Chapter of the association for computational linguistics: Human language technologies, Volume 1 (Long and Short Papers) (pp. 4129–4138).

Honnibal, M. \& Johnson, M. (2015, September). An improved non-monotonic transition system for dependency parsing. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1373–1378).

Islam, M. M. \& Hoque, A. L. (2010, December). Automated essay scoring using generalized latent semantic analysis. In 2010 13th International conference on computer and information technology (ICCIT) (pp. 358–363). IEEE.

Jirasatjanukul, K., Nilsook, P., & Wannapiroon, P. (2019). Intelligent human resource management using latent semantic analysis with the internet of things. *International Journal of Computer Theory and Engineering, 11*(2).

Kakkonen, T., Myller, N. \& Sutinen, E. (2006). Applying part-of-speech enhanced LSA to automatic essay grading. arXiv preprint cs/0610118.

Kanejiya, D., Kumar, A. \& Prasad, S. (2003). Automatic evaluation of students' answers using syntactically enhanced LSA. In Proceedings of the HLT-NAACL 03 workshop on building educational applications using natural language processing (pp. 53–60).

Khurana, D., Koli, A., Khatter, K. \& Singh, S. (2017). Natural language processing: State of the art, current trends and challenges. arXiv preprint arXiv:1708.05148.

Kuechler, W. L. (2007). Business applications of unstructured text. *Communications of the ACM, 50*(10), 86–93.

Kuncoro, A., et al. (2018). LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 1426–1436. In press.

Landauer, T. K., & Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review, 104*(2), 211.

Landauer, T. K., McNamara, D. S., Dennis, S. \& Kintsch, W. (Eds.). (2007). Handbook of latent semantic analysis. Lawrence Erlbaum Associates Publishers.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to latent semantic analysis. *Discourse Processes, 25*, 259–284.

Landauer, T. K., & Dumais, S. (2008). Latent semantic analysis. *Scholarpedia, 3*(11), 4356.

Landauer, T. K. (2002). Applications of latent semantic analysis. Proceedings of the annual meeting of the cognitive science society, 24.

Lenhard, W. (2008). Bridging the gap to natural language: A review on intelligent tutoring systems based on latent semantic analysis.

Liddy, E. D. (2001). Natural language processing.

Leacock, C., et al. (2010). Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 1–134. In press.

Linzen, Tal, Emmanuel, Dupoux, & Yoav, Goldberg (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 521–535. In press.

Mezher, R., & Omar, N. (2016). A hybrid method of syntactic feature and latent semantic analysis for automatic arabic essay scoring. *Journal of Applied Sciences, 16*(5), 209.

Mikolov, T., Chen, K., Corrado, G. \& Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mohler, M. \& Mihalcea, R. (2009, March). Text-to-text semantic similarity for automatic short answer grading. In Proceedings of the 12th conference of the European chapter of the ACL (EACL 2009) (pp. 567–575).

Nivre, J. (2005). Dependency grammar and dependency parsing. *MSI Report, 5133* (1959), 1–32.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*.

Pereira, J., & Díaz, Ó. (2019). What matters for chatbots? Analyzing quality measures for facebook messenger's 100 most popular Chatbots. In *Towards integrated web, mobile, and IoT technology* (pp. 67–82). Cham: Springer.

Schone, P. \& Jurafsky, D. (2000). Knowledge-free induction of morphology using latent semantic analysis. In Proceedings of the 2nd workshop on learning language in logic and the 4th conference on Computational natural language learning (Vol. 7, pp. 67–72). Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D. \& Singer, Y. (2003, May). Feature-rich part-of-speech tagging with a cyclic dependency network. In Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology (Vol. 1, pp. 173–180). Association for Computational Linguistics.

Vrana, S. R., Vrana, D. T., Penner, L. A., Eggly, S., Slatcher, R. B., & Hagiwara, N. (2018). Latent semantic analysis: A new measure of patient-physician communication. *Social Science & Medicine, 198*, 22–26.

Wang, Y. \& Zhao, H. (2015, October). A light rule-based approach to English subject-verb agreement errors on the third person singular forms. In Proceedings of the 29th Pacific Asia conference on language, information and computation: Posters (pp. 345–353).

Wegba, K., Lu, A., Li, Y. and Wang, W. (2018, March). Interactive Storytelling for Movie Recommendation through Latent Semantic Analysis. In 23rd International conference on intelligent user interfaces (pp. 521–533).

Wiemer-Hastings, P. \& Zipitria, I. (2001). Rules for syntax, vectors for semantics. In Proceedings of the annual meeting of the cognitive science society (Vol. 23, No. 23).

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. \& Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in neural information processing systems (pp. 5754–5764).

Yih, W. T., Zweig, G. \& Platt, J. C. (2012, July). Polarity inducing latent semantic analysis. In Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning (pp. 1212–1222). Association for Computational Linguistics.

Young, P., Lai, A., Hodosh, M., & Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics, 2*, 67–78.