Information Extraction for Knowledge Base Construction in the Music Domain

Abstract

Music content creation, publication and dissemination has changed dramatically in the last few decades. The rate at which information about music is being created and shared on the web is growing exponentially, which opens the challenge to make sense of all this data. In this paper, we present and evaluate a Natural Language Processing pipeline aimed at the learning of a Music Knowledge Base entirely from scratch. Our approach starts off by collecting thousands of "song tidbits" from the *songfacts.com* website. Then, we combine a state-of-the-art Entity Linking tool and a linguistically-motivated rule-based algorithm to extract semantic relations between pairs of entities. Relations with similar semantics are then grouped in semantic clusters by exploiting syntactic dependencies in relation patterns. Finally, a novel confidence measure over the set of extracted relations is introduced as a refinement step. Evaluation is carried out intrinsically, by assessing each component of the pipeline, as well as in an extrinsic task, namely Music Recommendation. An important contribution of our method is its ability to discover in text novel facts with high precision, which are missing in current generic and music-specific knowledge repositories. We release the datasets generated with our pipeline, together with the evaluation data, for the use and scrutiny of the community.

Keywords: relation extraction, entity linking, knowledge base creation, music recommendation, semantic web

2010 MSC: 00-01, 99-00

1. Introduction

Knowledge Representation and Reasoning is a key enabler of Intelligent Systems [1], and plays an important role in areas like Agents, Semantic Web, Video Understanding, or Natural language Understanding (NLU) [2].

In this paper, we focus on an important aspect of NLU, which is how to make sense of the data that is generated and published online on a daily basis. This data is mostly produced in human readable format, which makes it unsuitable for automatic processing. Considering that deep understanding of natural language by machines seems to be very far off [3], there is great interest in formalizing unstructured data, and Knowledge Bases (KBs) are a paradigmatic example of large-scale content turned into machine readable format.

We may define a KB as a repository of knowledge organized in a predefined taxonomic or ontologic structure, potentially compatible with other KBs, thus contributing to the Open Linked Data initiative[1]. These KBs may be designed to represent unsconstrained knowledge, or a single domain of interest. This representation is formalized either manually, automatically, or with a combination of both.

Regarding manually curated KBs, one of the best known is WordNet [4], a semantic lexicon which clusters together lemmas with equivalent meanings in "synonym sets" or synsets. It is however more frequent to find manually constructed KBs in restricted domains, e.g. Chemistry (*CheBi*[2]) [5], Genetics (*GeneOntology*[3]) [6], Medicine (*Snomed*[4]) [7], or Music (*MusicBrainz*[5]) [8]. The main reason for this being that modelling a constrained domain of knowledge is less open to *several equally correct alternatives*, since the degree of ambiguity is lower. However, and parting ways from domain-specific or manually generated KBs, the AI and NLP communities are showing a growing interest in

[1] http://linkeddata.org/
[2] https://www.ebi.ac.uk/chebi/
[3] http://geneontology.org/
[4] http://browser.ihtsdotools.org
[5] http://musicbrainz.org/

general-purpose KBs, unconstrained enough to potentially fit any domain, and with applications in knowledge-intensive areas such as Semantic Querying [9], Machine Translation [10], Vector Space Representations of Words and Senses [11, 12, 13, 14], Question Answering [15], Entity Retrieval [16], Recommender Systems [17], and Recommendation Interpretation [18, 19].

There is wide agreement in that this generic knowledge modeling can only be attained automatically, as it would not be sustainable to model manually all existing knowledge in all domains. Automatically generated KBs may be derived from structured data in collaborative resources (e.g. DBpedia [20]), from web crawling combined with truthfulness or reliability measures (e.g. NELL [21] or PATTY [22]), or from providing seamless integration of an unconstrained number of resources (e.g. BabelNet [23], or KB-Unify [24]).

Beyond domain-specific KBs, like Medicine or Chemistry, additional knowledge areas have been so far neglected in terms of automatic KB learning, one of them being the Music domain. There seems to be an agreement among Music Technology and Music Information Retrieval communities that there is an unfulfilled need for up-to-date, large scale Music specific datasets. In this paper, we aim to fill this gap by putting forward an NLP pipeline designed to construct a Music KB (MKB) entirely from scratch in an automatic and unsupervised manner. Starting from a large collection of documents in the Music domain, obtained from *songfacts.com*, we generate a fully disambiguated MKB with entity mappings against DBpedia, a general purpose resource, as well as MusicBrainz, a large database of music metadata. All relations have a relation pattern derived from a Relation Extraction procedure backed up by an algorithm which performs the following steps: (1) Morpho-syntactic rule-based *filtering*; (2) Dependency-based *clustering*; and (3) Relation *weighting* based on statistical evidence. We accompany our data with an intrinsic evaluation carried out on each component, as well as an extrinsic evaluation which consists of a *Music Recommendation* experiment where our automatically learned MKB is used to provide explanations to song recommendations in *natural language*.

Our experimental results indicate that our system is able to extract *high*

quality relations (Precision >= 0.8) as well as *novel knowledge*. We unveil thousands of relations absent in both large-scale generic KBs, as well as in Music specific resources. Moreover, the recommendation experiment shows that explanations based on the newly learned KB has a positive impact in music recommendation, enhancing it with meaningful explanations.

We release several versions of our KBs together with the evaluation data used in the experiments described in this paper. The code for the complete relation extraction pipeline is also released as open source.

2. Related work

The work described in this article strongly focuses on the exploitation of linguistic and semantic properties for the automatic learning of a MKB. For this reason, we deem relevant to cover related work in the following areas: (1) KB learning and curation, with special focus on Relation Extraction methods; and (2) The current state of KB learning and its applications in the Music domain.

2.1. KB Learning and Curation

We understand language by making sense of the connections between words, concepts, phrases and thoughts [25]. KBs constitute a resource for encapsulating this knowledge. Previous efforts on KB construction may be characterized as: (1) Hand-crafted KBs; (2) Integrative projects (automatic in design, but reliant on manually validated data); and (3) Fully automatic (also in the Relation Extraction process).

Among the first group, the best known is probably WordNet [4], a lexical database which groups concepts in "synonym sets", and encodes predefined relations among them such as *hyponymy/hypernymy*, *meronymy*, *holonymy*, or *instantiation*. Manually validated KBs, however, are mostly seen in specific domains, where the degree of ambiguity is lower and there is more availability of trained knowledge engineers.

Next, integrative projects are probably the most productive, as they are the largest projects in terms of content coverage and community involvement, not only users, but also contributors. Some examples include YAGO [1], an automatically created KB derived from integrating WIKIPEDIA and WORDNET; DBPEDIA [20], a collaboratively maintained project aimed at exploiting information present in WIKIPEDIA, both structured and in free text; FREEBASE [26], also a collaborative effort mainly based on extracting structured knowledge from WIKIPEDIA; or BABELNET [23], a semantic network which started as a seamless integration of WIKIPEDIA and WordNet, and today constitutes the largest multilingual repository of words and senses.

Concerning the third group we refer to approaches where knowledge is obtained automatically. Usually, these are framed within the *Open Information Extraction* (OIE) paradigm [27], which can be (roughly) summarized as (1) reading the web, (2) learning facts, (3) scoring them; and (4) structuring them according to some semantic criterion. Endeavours in this area include TEXTRUNNER [28], widely regarded as the first OIE system; REVERB [29], particularly designed to reduce noise while keeping a wide coverage, thanks in part to a set of syntactic and lexical constraints; NELL [30], which introduces semantic knowledge in the form of a hand-crafted taxonomy of entities and relations; PATTY [22] and WISENET [31, 32], in which a shared vision to integrate semantics is applied both at the entity and relation level; DEFIE [33], a recent development in OIE tested on the whole set of BABELNET glosses; and KB-UNIFY [24], not an actual OIE implementation, but rather a unification framework for OIE systems.

Another key aspect of automatic KB generation, in addition to semantics, is *how relations between entities are captured*. From rule-based linguistic approaches to state-of-the-art supervised machine learning methods, the general trend seems to attempt to extract as many facts as possible, with as much accuracy as possible, and keeping the degree of supervision low.

Previous work exploited combinations of surface and part-of-speech patterns [27] or regular expressions [29], as well as rules based on shallow parsing [31].

Furthermore, there are a number of contributions exploiting syntactic information in the form of syntactic dependencies, a linguistic formalism [34] that represents sentences as trees where each relation is bi-lexical and non phrasal, and where in general more important words (subject, verb, or direct object) appear higher in the tree. Syntactic dependencies have been extensively used in Relation Extraction, e.g. in supervised machine learning settings for computational lexicography [35]. Syntactic dependencies have played a role also in linking entity mentions, e.g. exploiting shortest paths between named entities [36], smallest spanning subtree covering two entities [37], or as part of a rule-based OIE system [38]. More recent work also benefited from dependency parsing, [22, 32, 24], where paths in the parse tree between two already identified entities were leveraged.

2.2. Music Knowledge Bases

In the music field, MusicBrainz and Discogs[6] are two examples of attempts to formalize knowledge. They are open Music encyclopedias of music metadata built collaboratively and available to the public. MusicBrainz, in addition, is regularly published as Linked Data by the LinkedBrainz project[7].

Generic resources like Wikipedia include a sizable amount of Music data, such as artist, album and song biographies, definitions of musical concepts and genres, or articles about music institutions and venues. By extension, KBs based on Wikipedia like the ones described earlier also include this information, and in a structured way. However, their coverage is limited as they are not specifically targeting the music domain, and they may miss novel and independent artists, albums or songs, and also Music works that are only locally relevant. Finally, let us refer to Grove Music Online[8], a Music encyclopedia containing over 60k articles written by Music scholars. However, this encyclopedia is not freely open and runs by subscription.

[6] `http://www.discogs.com`

[7] `http://linkedbrainz.org/`

[8] `http://www.oxfordmusiconline.com`

In addition to the aforementioned curated repositories, to the best of our knowledge, there is no an automatically learned and open MKB. A first step in this direction was taken in [39, 40], applying Relation Extraction techniques to big datasets of music related texts extracted from the web. In [41], a Flamenco MKB is created by combining information from curated KBs with information extracted from blogs and websites.

Despite their scarcity, MKBs are becoming increasingly popular in Music Information Retrieval (MIR) applications, such as artist similarity and music recommendation [42, 43, 44, 45]. MKBs have been also exploited as sources of explanations in Music Recommendation Systems. According to [46], giving explanations of the recommendations provides transparency to the recommendation process and increases the confidence of the user in the system. In [47], explanations of recommendations are created by exploiting DBpedia's structured information, whilst in [39], explanations are based on an automatically learned MKB.

3. Methodology

We propose a comprehensive method which learns a full-fledged Music KB entirely from scratch. In this paper, we report experiments after compiling our raw data from the Songfacts[9] website (see Section 4.1). This is a well suited resource both for KB learning and as a testebed for Relation Extraction due to its specificity.

Let us introduce the notation that will be referenced throughout the description of the method. First, we denote our KB *Vocabulary* as the set $\mathcal{V}$, which comprises the following:

- $\mathcal{E} = \{e_i, e_j, ...e_n\}$ Set of disambiguated entities, e.g. {*Born in the USA*$_{dbp}$[10], *Bruce Springsteen*$_{dbp}$, ... }.

[9] `http://www.songfacts.com`

[10] We use the *dbp* subscript to refer to disambiguated entities linked to DBpedia.

- $\Upsilon = \{\upsilon_i, \upsilon_j, ...\upsilon_n\}$ Set of entity types, e.g. {*Album*, *MusicalArtist*, ... }.
- $\mathcal{P} = \{p_i, p_j, ...p_n\}$ The set of relation patterns, e.g. {*was recorded by frontman*, ... }.
- $\mathcal{C} = \{c_i, c_j, ...c_n\}$ Set of cluster patterns, e.g. {*was recorded by, ...*}.

$\mathcal{R}$ defines the set of all extracted relations that will be included the KB. Every $r \in \mathcal{R}$ is defined by $(\boldsymbol{e_d}, \boldsymbol{e_r}, \boldsymbol{\upsilon_d}, \boldsymbol{\upsilon_r}, \boldsymbol{p}, \boldsymbol{c})$, where $\boldsymbol{d}$ and $\boldsymbol{r}$ refer to the relation domain and range respectively. From a relation $r \in \mathcal{R}$, a triple $\langle d, rel, r\rangle$ may be derived provided that $d, r \in \{\mathcal{E} \cup \Upsilon\}$ and $rel \in \{\mathcal{P} \cup \mathcal{C}\}$. We list some prototypical triples available from the above sets.

- $t_{\boldsymbol{p}} : \langle \boldsymbol{e_d}, \boldsymbol{p}, \boldsymbol{e_r}\rangle$, e.g. {*Born in the USA*$_{dbp}$ - *was recorded by frontman* - *Bruce Springsteen*$_{dbp}$}.
- $t_{\boldsymbol{c}} : \langle \boldsymbol{e_d}, \boldsymbol{c}, \boldsymbol{e_r}\rangle$, e.g. {*Born in the USA*$_{dbp}$ - *was recorded by* - *Bruce Springsteen*$_{dbp}$}.
- $\tau_{\boldsymbol{p}} : \langle \boldsymbol{\upsilon_d}, \boldsymbol{p}, \boldsymbol{\upsilon_r}\rangle$, e.g. {*Album* - *was recorded by frontman* - *MusicalArtist*}.
- $\tau_{\boldsymbol{c}} : \langle \boldsymbol{\upsilon_d}, \boldsymbol{c}, \boldsymbol{\upsilon_r}\rangle$, e.g. {*Album* - *was recorded by* - *MusicalArtist*}.

Finally, different subsets of $\mathcal{R}$ can be constructed by doing a selection of its tuples:

- $\mathcal{R}_p = \{r_1^p, ...r_n^p\}$ All relations with a specific relation pattern p.
- $\mathcal{R}_c = \{r_1^c, ...r_n^c\}$ All relations with a specific cluster pattern c.
- $\mathcal{R}_{\boldsymbol{\tau_p}} = \{r_1^{\tau_p}, ...r_n^{\tau_p}\}$ All relations with a specific relation pattern, domain and range.
- $\mathcal{R}_{\boldsymbol{\tau_c}} = \{r_1^{\tau_c}, ...r_n^{\tau_c}\}$ All relations with a specific cluster pattern, domain and range.

We can thus define a KB as a tuple $(\mathcal{V}, \mathcal{R})$, where at initialization stage, $\mathcal{V} = \emptyset$, $\mathcal{R} = \emptyset$. In what follows, we describe a computational method that

acquires new entities, types and relations and combines them in a meaningful way to enrich the KB.

3.1. Morphosyntactic Preprocessing

Our morphosyntactic preprocessing module takes as input a collection of text documents in the Music domain. First, sentence splitting and tokenization is carried out thanks to the *Stanford NLP tokenizer*[11]. Next, a dependency parse tree is obtained via the MATE Parser, described in [48]. We justify the use of this tool because of the richness of the tagset of its training data and hence, the possibility to craft fine-grained rules over dependency relations.

In a dependency tree, each node includes information, at least and depending of the model and the language, about surface and lemmatized forms, along with its part-of-speech. Each edge in the tree is labeled with a dependency relation such as *subject* or *noun modifier* (an example is shown in Figure 1).

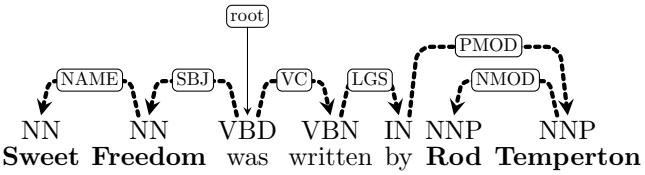


Figure 1: Example sentence with dependency parsing tree

3.2. Semantic Processing: Entity Linking

Entity Linking acts as a semantic bridge between plain text and a reference knowledge inventory. In our specific case, we are not only interested in identifying mentions of entities, but to map these mentions to the dependency representation of the sentence.

While there are a number of popular Entity Linking systems which are not bound to any domain or discipline, there is no benchmark of such systems in the Music domain. Therefore, we do not know *a priori* how well each of them works in Music corpora. Musical entities may raise a plethora of challenges,

[11] http://nlp.stanford.edu/software/tokenizer.shtml

derived mostly from ambiguity and polysemy. For example, an album may have the same name as the band who recorded it (e.g. *Weezer* the band and their first album). An artist, a song or an album may have words or expressions much more common in another domain or area of knowledge, which may constitute a source of confusion (e.g. *Berlin, The Who*). Thus, the choice of the best Entity Linking algorithm or off-the-shelf tools is crucial, as potential errors may propagate throughout the different modules and hinder considerably the quality of the resulting KB.

Among the available state-of-the-art systems we considered, Tagme, Babelfy and DBpedia Spotlight [49], we opted for the latter, as it has shown to be the least prone to errors in musical texts (further details are provided in Section 5.1).

3.2.1. Adding Co-references

In the Music domain, typical text resources such as artist biographies, album reviews, or song tidbits, are normally tied to a specific entity. Based on this evidence, we exploit any metadata associated to a source document (e.g. an artist's name), co-referential pronouns and *resource-specific coreferences*, i.e. implicit but idiosyncratic mentions of an entity. We proceed as follows. For a given entity reported in a document, we retrieve its corresponding URI from a reference KB (MusicBrainz and DBpedia in the particular case of the Songfacts dataset). Then, pronouns are replaced by the entity title and disambiguated with the URI of the entity they unequivocally refer to.

A similar approach is used in [18], where the frequency of pronouns "he" and "she" is computed in every document to determine the entity's gender, and then, these pronouns are replaced by the entity title. Similarly, in [40], a gender identifier web service is used to determine the gender of subjects in artist biographies as part of a Relation Extraction pipeline. In addition, we have observed an exploitable *resource-specific coreference* in Music reviews, where terms like "this album" or "the song" can be replaced by the document's title. In the dataset used for the experiments (see Section 4.1), the expressions"this song" and "the song" are replaced with the name of the song as it appears in the docu-

ment, and disambiguated with the URI of the entity they unequivocally refer to. Finally, exact string matches of these songs mentions are also straightforwardly annotated in the document.

3.2.2. Type Filtering

In DBpedia, most resources are associated with one or more types via the `rdf:type` property. In addition, among the different types present in DBpedia (coming from the DBpedia ontology, Yago types, or `schema.org`), the DBpedia ontology provides a relatively small and tidy taxonomy of 685 classes based on Wikipedia infoboxes. Other KBs such as Yago or Freebase have their own classification criteria, which are larger but noisier. MusicBrainz, in contrast, has a very narrow set of entity types.

This type information can be exploited in order to narrow down the set of permitted types for a given candidate and its potential annotations. In this way, we ensure that all entities will be, at least, related to the music domain. Restricting the search space to types such as Artist or Song reduces considerably the number of errors derived from cross-domain ambiguity.

Depending on the envisioned application of the KB resulting from our pipeline, the predefined set of entity types may vary. In our case we restricted them to Musical Artists, Other Artists, Songs, Albums, Genres, Films and Record Labels. We considered including other types such as Place, Event, Award or Broadcaster, but results were not satisfactory, as they were too broad and, too often, imprecise. In Table 1 we present the mapping between the DBpedia ontology, MusicBrainz entity types and our selected set of types.

After the Entity Linking process is complete, $\mathcal{V} = \mathcal{V} \cup \{\mathcal{E}, \Upsilon\}$.

3.3. Syntactic Semantic Integration

The information obtained from the syntactic and semantic processes is combined into a graph representation of the sentence. For each Music entity identified during the semantic enrichment step (Section 3.2), all nodes in the dependency tree with a correspondence to an entity mention are collapsed into

Our MKB	DBpedia ontology	MusicBrainz
MusicalArtist	Person/Artist/MusicalArtist Organization/Band Writer/MusicComposer Writer/SongWriter	Artist
OtherArtist	Person/Artist (¬ MusicalArtist) Person/Writer(¬ MusicComposer & ¬ SongWriter)	—
Album	Work/MusicalWork/Album	Release
Song	Work/MusicalWork/Song Work/MusicalWork/Single	Recording Work
Genre	TopicalConcept/Genre	—
Film	Work/Film	—
RecordLabel	Agent/Organization/Company/RecordLabel	Label

Table 1: Type mapping

one single node: *Sweet* and *Freedom* into *Seet Freedom (Album)*, and *Rod* and *Temperton* into *Rod Temperton (Artist)*. Figure 2 shows the resulting syntactic-semantic representation of a sentence.

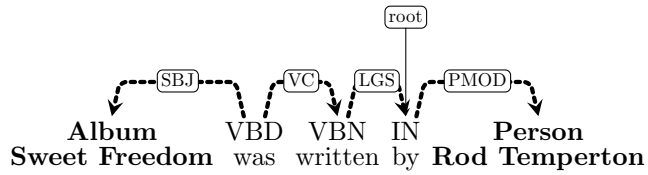


Figure 2: Semantic integration on syntactic dependencies

3.4. Relation Extraction

Having syntactic and semantic information available, potential relations between entities can be discovered by traversing the dependency tree. Two entities in such tree are deemed related if there is a path between them that does not contain any other entity in between, and does not contain parentheses. If there is more than one path we consider only the shortest path as the representative path of the relation.

Our method encodes a relation pattern between two entities as all words in the shortest path between them. In the example provided in Figure 2, the shortest path between *Sweet Freedom* and *Rod Temperton* contains the words

was, *written* and *by*. The KB enrichment derived from the Relation Extraction process can be formally expressed as follows: $\mathcal{V} = \mathcal{V} \cup \{P\}$, and every extracted relation $r \in \mathcal{R}$ is expressed as $r = \{e_d, e_r, \upsilon_d, \upsilon_r, p\}$.

3.5. Filtering

Relation Extraction via shortest path in syntactic trees is common practice in the literature [33]. However, not all shortest paths are valid, and incorrections may be derived of overly long and syntactically complicated sentences, or the use of reported speech. We surmount these problems by defining three filtering heuristics over surface forms (*lemma-paths*), part-of-speach patterns (*pos-paths*), and labels of syntactic dependencies (*dependency-paths*).

First, we filter out all relations with reporting verbs (e.g. "say", "tell" or "express") in the lemma-path. The intuition being that sentences with these verbs are syntactically complex and relations in them may not be encoded in shortest paths, as in the following sample sentence, where the relation is incorrect, and hence would be pruned out:

> **Sentence: Nile Rodgers** *told* NME that the first album he bought was Impressions by **John Coltrane**.
>
> **Relation**: **nile_rodgers**$_{dbp}$ told that was impressions by **john_coltrane**$_{dbp}$

Second, we only selected relations whose dependency-path starts with a subject (which may also preceded by a nominal modifier or an apposition), a direct or indirect object, a predicative complement or a verb chain. When this condition holds, the relation is considered *good*. If the above condition does not hold, an extra validation step is applied over the pos-path in order to capture relations without verbs, which seem to be highly idiosyncratic of the music domain, e.g. $\langle e_d, \texttt{frontman of}, e_r \rangle$, $\langle e_d, \texttt{drummer}, e_r \rangle$, or $\langle e_d, \texttt{guitarist and singer}, e_r \rangle$.

The output obtained after the filtering process is called $\mathcal{R}^*$, where $\mathcal{R}^* \subset \mathcal{R}$. $\mathcal{R}^*$ contains those relations $r \in \mathcal{R}$ which successfully pass the filtering stage.

3.6. Dependency-Based Loose Clustering

In this section we describe a simple but powerful clustering algorithm aimed at reducing noise in our relation patterns inventory $\mathcal{P}$.

Let us consider the following three relation patterns: (1) *was written by blunt producer*, (2) *was written by singer/producer*, and (3) *was written by manager and guitarist*. Intuitively, these three relation patterns seem to be semantically similar, and if all of them were expressed as *was written by*, the original meaning would not be lost, and the set of relations would become much more compact.

This observation, which we found to occur quite frequently, motivated the inclusion of a *dependency-based loose clustering* module. First, we perform a second run of dependency parsing over all $p \in \mathcal{P}$ aiming at discovering the root node of the relation pattern linking to previously identified entity mentions. Note that the root of the original sentence does not need to correspond with p's root. Then, our algorithm considers all possible paths from the root to every leaf node of the dependency tree, and selects the path that complies with a predefined syntactic constraint (e.g. verb chain plus adverb or preposition, adverb plus nominal and preposition modifiers) based on regular expressions of syntactic labels. The sequence of tokens that matches this regular expression constitutes the cluster pattern. Note that a cluster pattern does not necessarily need to match a full path from root to leaf node.

As an illustrative case, consider the extracted relation pattern "is track was released on label". After parsing, we obtain the parse tree shown in Figure 3 and a cluster pattern (in bold) over those nodes in the dependency tree that satisfy one of the regular expressions crafted in the aforementioned syntactic constraint.

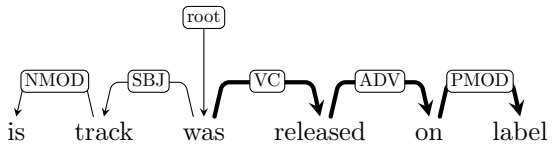


Figure 3: Example of a parsed relation pattern $p \in \mathcal{P}$. In column three, sX refer to Song entities, maX to MusicalArtist and aX to Album.

Filtering out spurious information in OIE following similar approaches has proven effective while not being computationally expensive [29]. Ours is a *loose clustering* method because it enforces a pattern to *partially* match at least one *regexp* rule, however this rule does not need to apply to the whole relation pattern.

The clustering stage provides an enrichment of all $r \in \mathcal{R}$ such that $r = r \rightarrow c$, where c is the cluster pattern derived from the relation pattern p. A relation cluster is the set of relations with the same cluster pattern, and is denoted as $\mathcal{R}_c$. Finally, the *Vocabulary* is enriched such that $\mathcal{V} = \mathcal{V} \cup \{\mathcal{C}\}$.

Cluster pattern c	**Typed cluster pattern** τ_c	**Relation triples** t_p
was written by	S *was written by* MA	s1 *was writen by artist* ma1
		s2 *was written by composer* ma2
		s3 *was written by singer* ma2
		s4 *was writen by* ma1
		s5 *was written by frontman* ma3
	A *was written by* MA	a1 *was written by frontman* ma3
		a2 *was written by guitarist* ma1
		a3 *was written by artist* ma2
		a4 *was written by frontman* ma5

Table 2: Example of a relation cluster $\mathcal{R}_c$, where $c =$ *was written by*.

3.7. Scoring

So far, our approach has identified entity mentions in text and has linked them in meaningful relations, filtering out those that did not comply with predefined linguistic rules. We incorporate one additional factor $score(r)$ that takes into account statistical evidence computed over $\mathcal{R}$. It has three main components, which we flesh out as follows:

We hypothesize that the relevance of a cluster may be inferred by the number and proportion of triples it encodes, and whether these are evenly distributed. Our metric encompasses a combination of three different components. First, we focus on the *degree of specificity* of the relation cluster, as previous work has demonstrated that this can contribute to Information Extraction pipelines [24]. Second, we analyze *intrinsic features* of the relation pattern, such as frequency,

length and fluency. Finally, we incorporate a *smoothing factor*, namely the proportion of the related typed pattern in the cluster.

A cluster $\mathcal{R}_c$ can be decomposed into a set of typed cluster patterns τ_c (see Table 2). The intuition behind the specificity measure of a cluster is that clusters with a highly predominant τ_c are more specific, i.e. they are largely used for encoding one specific type of relations. One example of this would be *performed with*, which enforces a relation to include MusicalArtists on both the domain and range sides. Thus, we define $\mathcal{L}_c$ as the list of cardinalities (number of triples) of every typed cluster pattern $\tau_c \in \mathcal{R}_c$, being $\mathcal{L}_c = \{|\mathcal{R}_{\tau_c^1}|, ..., |\mathcal{R}_{\tau_c^n}|\}$. We define the specificity measure as the variance of $\mathcal{L}$, expressed as $s(\mathcal{R}_c) = \sigma^2_{\mathcal{L}_c}$.

Furthermore, the notion of *relation's fluency* is aimed at capturing its comprehensibility. Simply put, the more the original word order is preserved in the relation pattern, the more understandable it is. This is introduced due to the fact that word order is lost after modelling text under a dependency grammar framework. We modelled this as a *penalty measure* over the number of jumps needed to reconstruct the original order. Let k be the number of tokens in the relation pattern, w_i the $i-th$ word in the pattern and $h(w_i)$ a function that returns the correspondent word index in the original sentence, we put forward a fluency measure f defined as:

$$f(p) = \frac{\sum_{i=1}^{k} \alpha |h(w_i) - h(w_{i-1})|}{k} \tag{1}$$

where $\alpha = 2$ if $h(w_{i-1}) > h(w_i)$ and $\alpha = 1$ otherwise. Note that higher values of f means low fluency. For instance, for the relation pattern *is hit for* the score would be much higher than a mixed-up order relation pattern such as *joined because added were and hit*, would have a very high f.

Finally, the confidence measure for each relation $r \in R$ is expressed as follows:

$$score(r) = \left(s(\mathcal{R}_c) + \frac{|\mathcal{R}_p|}{|p| + 2^{f(p)}} \right) \times \frac{|\mathcal{R}_{\tau_c}|}{|\mathcal{R}_c|} \tag{2}$$

As an illustrative example of the measure, the score of a relation whose

typed pattern is ⟨ Song, *was released on*, RecordLabel ⟩, will have a much higher score than a relation whose typed pattern is ⟨ Album, *was released on*, MusicalArtist ⟩. This latter pattern is obviously incorrect, and it is probably due to a disambiguation error in the Entity Linking step. Thus, the statistical measure is helping the system by discriminating good and bad relations.

4. Experimental Setup

In this section, we describe the setting under which the experiments are carried out. We refer, first, to the source corpus, and second, to the resulting KBs as output of different subsets of our approach.

4.1. Source dataset

Songfacts[12] is an online database that collects, stores and provides facts, stories and trivia about songs. These are collaboratively written by registered users, and reviewed by the website staff. It contains information about more than 30.000 songs from nearly 6.000 artists. This information may refer to what the song is about, who wrote it, who produced it, who collaborated with whom or who directed the video clip. It stems from the above that these texts are rich sources of information not only well-known Music facts, but also more specific trivia, as in the following sample sentence (about David Bowie's *Space Oddity*): "Bowie wrote this song after seeing the 1968 Stanley Kubrick movie 2001: A Space Odyssey".

We crawled the Songfacts website in mid-January 2014. Then, for each song article, we performed a mapping between the song and its MusicBrainz song ID, using the MusicBrainz Search API. We successfully mapped 27,655 songs.

The Relation Extraction pipeline was run over the 27,655 document Songfacts corpus, which amounts to 306,398 sentences. After the Semantic Processing step, we obtained 202,767 entity mentions (8,880 for *album*, 3,136 *record labels*, 74,908 *songs*, 107,253 *musical artists*, 1,760 *genre* labels, 3,467 for *other*

[12] http://www.songfacts.com

artist, and 3,363 for *film*). There were 48,122 sentences with at least two entities, and it is on this subset where we apply our Relation Extraction pipeline.

4.2. Learned Knowledge Bases

Our aim is to assess to what extent each of the modules integrating our approach contribute to the quality of the resulting KB. After firing up the whole pipeline, we generate two *learned* KBs, two *baseline* KBs, and a *competitor* KB.

The *learned* KBs are the result of applying the Relation Extraction method to the Songfacts dataset under different conditions. KBSF-ft is derived from applying the Relation Extraction pipeline entirely, and KBSF-th comes from a selection of all triples in KBSF-ft with a confidence score above a certain threshold. In addition, we created two baseline KBs for evaluation purposes. KBSF-co is the baseline which consists of a simple entity co-occurrence. More specifically, if two entities are mentioned in the same sentence, a triple that anchors them is added to the KB with the predicate "related_to". In addition, KBSF-raw was created following the relation extraction pipeline, but without applying the filtering process described in Section 3.5. Finally, KBSF-rv constitutes the competitor KB, and is built as follows: After running ReVerb over the Songfacts dataset, we search coinciding relations, at both domain and range positions, that include entity mentions identified in our disambiguation step. These are included in KBSF-rv. Statistics about these five KBs are reported in Table 3.

KB	Entities	Triples	Relation Predicates	Cluster Predicates
KBSF-ft	20,744	32,055	20,438	14,481
KBSF-th	10,977	11,720	2,484	828
KBSF-co	30,671	113,561	1	—
KBSF-raw	29,280	71,517	47,089	32,712
KBSF-rv	9,255	7,532	2,830	—

Table 3: Statistics of all the learned KBs

To determine the best threshold to truncate KBSF-ft, we aimed at maximizing the number of triples and at the same time minimizing the number of

relation patterns. Our intuition is that less patterns means a tidier KB. We computed the percentage of triples and relation patterns from KBSF-ft that remain in a truncated KB, whose triples have a score greater than a threshold θ. We computed these percentages for every θ value ranging from 0 to 1 in steps of 0.01 (see Figure 4). Our goal was to reveal the θ value which maximizes the difference between the number of triples and the number of triple patterns in a truncated KB. After confirming a maximized difference with $\theta = 0.05$, we created KBSF-th where all triples in KBSF-th have a score greater or equal than θ. Statistics about the KBs described in this paper are shown in Table 3.

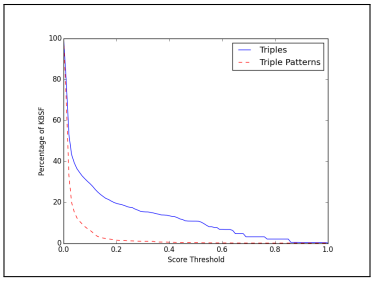


Figure 4: Percentage of Triples and Triple Patterns at different values of θ

5. Experiments

5.1. Quality of Entity Linking

We mentioned in Section 3.2 the lacking of both Music-specific Entity Linking tools as well as benchmarking datasets. For this reason, we performed a set of experiments to select the best suited Entity Linking tool for the Music domain, among the best known and reputed. Specifically, we perform evaluation experiments on DBpedia Spotlight, TagMe and Babelfy. Let us briefly describe each of them:

- **Babelfy** [50] A graph-based system for Entity Linking and Word Sense Disambiguation. It operates with BabelNet as a reference sense inventory.

- **Tagme** [51] An Entity Linking automatic annotator which matches terms with WIKIPEDIA hyperlink texts and disambiguates them using both the in-link graph and the page datasets. It incorporates a context-aware pruning step.

- **DBpedia Spotlight** [49] Automatically identifies entity mentions in free text, linking each match with its corresponding DBPEDIA URI.

We created a dataset of annotated musical entities and applied both quantitative and qualitative evaluations in order to confirm which system performs better with musical entities, and which is more suitable for our task.

5.1.1. Output Harmonization Procedure

As of now, most Entity Linking systems *speak their own language*, partially due to the fact that each of them is designed on the back of different KBs, and because their output is heterogeneous in format and cannot be directly compared, let alone combine.

In order to evaluate and derive a unified prediction integrating the three Entity Linking systems under consideration, we proceed as follows: First, we retrieve DBPEDIA URIs of every named entity. There are some considerations to be taken into account, namely: (1) Character encoding differs from system to system; (2) Several URIs may refer to the same DBPEDIA resource, which we solve thanks to the transitive redirections provided by DBPEDIA; and (3) In the specific case of Tagme, only WIKIPEDIA page IDs are provided, which we exploit to map entity mentions to their DBPEDIA equivalent. Finally, after surmounting compatibility issues among systems, we retrieved DBPEDIA types (rdf:type property) for all entities.

5.1.2. Evaluation Data

We created an *ad-hoc* gold standard dataset to evaluate the different systems. The dataset was created as follows, with the Songfacts dataset (Section 4.1) as our testbed. In this corpus, each document univocously refers to one single song.

In addition, we have information about artist and song names at our disposal. We used this information to obtain the MusicBrainz ID for songs and artists. In MusicBrainz, artist and song items sometimes have information about their equivalent Wikipedia page. We leveraged this information, when available, to obtain their correspondent DBpedia URIs. Finally, we obtained a mapping with DBpedia of 7,691 songs and 3,670 artists. From the DBpedia resources of each song, we gathered their corresponding album name and URI if available, obtaining information of about 2,092 albums. Then, for every document, we looked for exact string matches of the reported song, and its related album and artist names. Every detected entity is thus annotated with its DBpedia URI. At the end of this process, the newly created gold standard dataset contains 6,052 documents where 17,583 sentences are annotated with the following entities: 5,981 Songs, 12,137 Artists and 1,722 Albums. As mentioned in Section 3.2, there are typical cases of ambiguity in musical entities where songs, artists and albums can share the same name. Therefore, we manually corrected the entities detected in 212 documents where this kind of ambiguity was present.

	Album		Artist		Song		Macro Average		
	Prec	Rec	Prec	Rec	Prec	Rec	Prec	Rec	F-measure
Babelfy	0.93	0.28	0.98	0.55	0.96	0.31	0.96	0.38	0.54
Tagme	0.75	0.69	0.97	0.77	0.65	0.71	0.79	0.72	0.76
Spotlight	0.80	0.52	0.94	0.83	0.59	0.42	0.78	0.59	0.67

Table 4: Precision and recall of the Entity Linking Systems considered

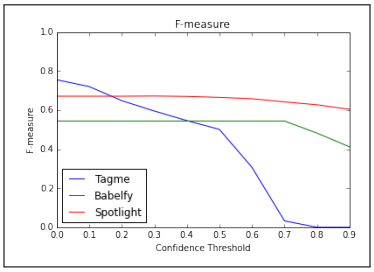


Figure 5: F-measure of the Entity Linking systems at different confidence thresholds

We evaluated the output of the three entity linking systems under review,

filtering out the entities with confidence measure below to a certain threshold Θ. We run the evaluation for different values of Θ, ranging from 0 to 0.9 in intervals of 0.1. After performing the evaluation over the gold dataset, the best results in terms of F-measure were obtained by all the systems at $\Theta = 0$ (see Figure 5). Detailed results on the best run of every system are shown in Table 4. We used macro-average Precision and Recall measures, i.e. we averaged their values from the three sets of entities.

We can conclude from the results that Babelfy is the system with highest precision on musical entities. However, its recall is quite low compared to the other systems, and specifically with Tagme, which in turns, shows much lower precision. DBpedia Spotlight, on the other hand, achieves a similar precision score than Tagme, but with a slightly lower recall.

This evaluation experiment is only focused on measuring the precision in the annotation of entities present in the gold standard. However, since all possible entities in a document may be not annotated, we also report on specific types of false positives which emerged during a qualitative inspection of classification results. For example, a frequent error that is not being evaluated concerns cases in which a text span not annotated in the gold is identified incorrectly as an entity by the tool. Therefore, to complement the evaluation, we listed the most frequently identified entities by each system (see Table 5). As we can see, Babelfy and Tagme are misidentifying common words as entities very frequently, whereas DBpedia Spotlight is not doing so. These errors may propagate to the rest of the Relation Extraction pipeline, and for this reason, despite Tagme showing better overall performance, we decided to use DBpedia Spotlight to feed our system with semantic annotations.

5.2. Quality of Relations

Relation Extraction evaluation is not trivial as semantic relations between entities may vary in terms of correctness over time. Also, correct relations may be linguistically flawed, i.e. not fluent. Previous approaches assessed automatically extracted relations in terms of correctness according to human judgement

System	Song	Album	Artist
Babelfy	**Carey**	**Debut**	John_Lennon
	Stephen	**Song_For**	Eminem
	Rap_Song	**Sort_Of**	Paul_McCartney
	Singing_This_Song	**First_Song**	Bob_Dylan
	A_Day_in_the_Life	**Debut_Album**	Drake
Tagme	**The_Word**	**Up!**	John_Lennon
	The_End	**When_We_On**	**The_Notorious_B.I.G.**
	If	**Up**	**Do**
	Once	**Together**	Paul_McCartney
	For_You	**By_the_Way**	**Neil_Young**
Spotlight	Sexy_Sadie	The_Wall	Madonna
	Helter_Skelter	Let_It_Be	Eminem
	Cleveland_Rocks	Born_This_Way	Rihanna
	Stairway_to_Heaven	Thriller	John_Lennon
	Minnie_the_Moocher	Robyn	Britney_Spears

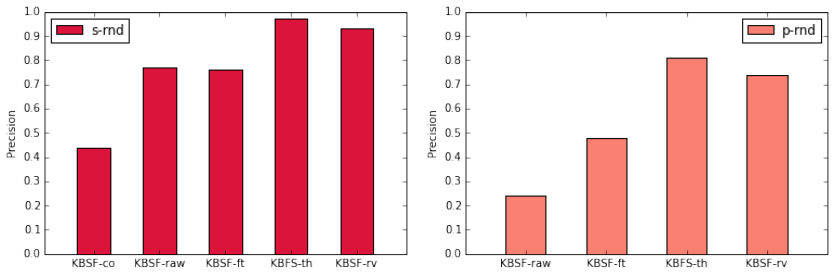
Table 5: Top-5 most frequent entities by type and tool. Disambiguation errors appear in bold.

[29, 52]. Additionally, a finer grained analysis is carried out in [28], adding a prior step in which relations are judged as being *concrete* or *abstract*.

In this paper, we made use of extensive human input and asked two experts in Computational Linguistics to evaluate the *top 100* scoring relations as yielded by our weighting policy (Section 3.7), as well as a random sample of 100 relations. This was done for all the KBs produced by our pipeline and for KBSF-rv.

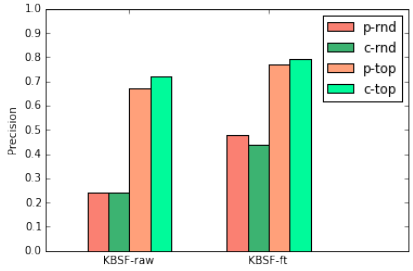
In Figures 6a and 6b, where we compare random samples from each KB, we observe a progressive improvement of the quality of relations as the different modules of our implementation kick in. The difference between these figures is that in the former, a relation is deemed correct if it has extracted a relation *expressed in the original sentence*, whereas the latter figure reports numbers on whether the extracted relation pattern was correct, i.e. if it *meant* the same as it was intended in the source sentence. We can infer from these results that co-occurrence between entities does not guarantee an explicit relation, whereas the presence of a path between two entities over a sentence dependency tree, without any other entity mention in between, generally suggests that we are in front of a monsemous and unambiguous relation between them.

It is remarkable how well ReVerb performs (Figure 6b), only being surpassed by the KB resulting from the complete implementation described in this paper. We note that the good results of the ReVerb extractor are also affected



(a) In sentence

(b) Relation patterns



(c) Patterns and clusters

Figure 6: Precision of relations at sentence, relation pattern and cluster pattern levels in top and random samples of relations

by the semantic processing of our system, which is selecting good candidates as relation arguments. Recall that the difference between KBSF-ft and KBSC-th is the inclusion of the *scoring* module, and the increase in Precision confirms that incorporating *statistical evidence contributes to better relations*.

This is further confirmed in the results showcased in Figure 6c, where we provide a comparison between top 100 relations according to our ranking policy against a random sample. Note that *in all* KBs*, highly scoring relations are more often marked as correct*, which constitutes additional support for the contribution of the novel weighting approach we introduced. Together with the quality of the relation pattern, this plot shows the quality of the cluster pattern associated with the evaluated relations. We observe that cluster patterns inferred in our clustering module have similar quality than relation patterns in the random sample, and slightly better in the top 100 sample. This results

implies that the scoring module is rewarding good clusters.

Finally, we computed Cohen's Kappa interannotator agreement over each evaluation set. It ranged from 0.60 to 0.81, which is generally considered as *substantial* agreement.

5.3. Coverage of the Extracted Knowledge Base

With this experiment we aim to compare the coverage of music relations in our KBs with other resources with human intervention, such as DBpedia, MusicBrainz, and with fully automatic resources. For the latter, we considered DefIE as our closest competitor due to several methodological similarities (dependency parsing, Entity Linking and relation extraction shortest path). To be able to compare the knowledge bases we need to have a set of relation instances whose domain and range entities can be mapped to entities in all of the knowledge bases of the comparative.

We selected all triples in KBSF-th whose domain and range entities could be mapped to both DBpedia and MusicBrainz. As our extracted KB has only MusicBrainz ID of entities of types MusicalArtist and Song, the set of triples to evaluate is restricted to relations between this kind of entities. Since entities in DefIE are disambiguated with BabelNet ids, we mapped all DBpedia uris to their corresponding BabelNet id, which yielded a subset of 3,633 triples. From here, we selected all possible domain-range entity pairs, and retrieved from the other KBs all triples with the same pairs, and counted them. The procedure to do so on DBpedia was via SPARQL queries.We discarded triples with predicate *wikiPageWikiLink*, as this predicate means an unlabeled relation. However, the mapping with MusicBrainz was not trivial. MusicBrainz is not a KB of triples, but a relational database. Entities are stored in tables, and relations between entities are represented in a set of tables of relations, having one table for each possible relation. The entities in the studied set of triples were only of type MusicalArtist and Song. However, an entity of type Song in KBSF-th can be related to either a Recording or a Work entity in MusicBrainz (see Section 3.2.2). Therefore, for the analysis of relations involving a Song entity,

we obtained the equivalent Recording and Work MusicBrainz entities, and looked for relations where any of them where present.

Mapping results are shown in Table 6. Let us highlight the fact that most semantic relations encoded in KBSF-th are novel, as they were not found in any of the other resources we compared against. In the overlap cases, most of the times the relation labels were semantically equivalent, and often the relation label of **KBSF**-th triples was more specific than the ones retrieved from other **KB**s (e.g. *frontman* and *member of*)

	KBSF-th	MusicBrainz	DBpedia	DefIE
Relation instances	3,633	1,535	1,240	456

Table 6: Number of triples with labeled relations in the different KBs for the same set of domain-range entity pairs

5.4. Interpretation of Music Recommendations

The main aim of this experiment is to evaluate the suitability of KBSF-th to explain relations between songs, and study their impact in the user's experience. Since our aim is not to measure the performance of a recommender system, we implemented a baseline recommender approach. Recommendations are based on a concept of song similarity which exploits the structure of our KB, following [43].

We designed the experiment as an online survey, where the participant is first asked to select 5 songs from different artists of his/her choice. From each selected song, the system randomly selects 3 recommendations among the list of its top-10 most similar songs. One of them is shown together with an explanation in natural language (the source text), another with an explanation based on relation patterns, and the third one has no explanation. All songs can be listened to thanks to an embedded player. After listening to the recommendation and reading any explanation attached to it, participants were asked to rate each recommendation from 1 to 5 (1 being worst), and to mention whether they were familiar or not with the recommended songs.

The experiment involved 35 participants, 28 males and 7 females, ranging from 26 to 38 years old and with different musical background and listening habits. Most of the participants affirmed that they had previous experience with recommendation systems. A total of 525 answers (corresponding to individual song recommendations) were collected. In 38% of the cases, the user was familiar with the recommended songs.

The average rating of recommendations with natural language explanations is slightly higher (3.20±1.29) than recommendations without explanations (3.08±1.35), or with explanations based on relation labels (3.04±1.34). In addition, for musically educated subjects, recommendations of unfamiliar songs, whether accompanied with or without explanations, have similar average rating (2.87 and 2.95 respectively). However, for untrained users, recommendations with explanations have a remarkable higher average rating (2.93) than without them (2.36). Thus, we can infer that the introduction of explanations in recommender systems improves the user experience of musically untrained subjects when discovering songs.

We also asked the subjects to select among a set of adjectives those that better described the recommendation experience. The general trend was to rate positively the experiment. Most users rated the experience as *enjoyable* (40%), followed by *useful* (31%) and enriching (29%). Negativity was much lower in general, with *confusing* being the most voted (17%), followed by *complicated* and *too geeky* (8% in both cases). This suggests that the introduction of explanations generated from our MKB in the recommendations was in general a satisfactory experience to users.

6. Conclusions and Future Work

We have presented an NLP pipeline that learns a Knowledge Base in the Music domain entirely from scratch. It combines methods easily applicable to a general purpose application with domain-specific heuristics which are designed to exploit particularities of the domain.

The result of our approach is a new Music Knowledge Base, which encodes semantic relations among musical entities. Our method relies on the syntactic structure (defined via dependency parsing) of sentences and the use and adaptation of Music-specific heuristics for both Entity Linking and Relation Extraction. In addition, we include modules for semantic clustering and pattern scoring, aimed at the efficient removal of noisy relations. Our modular evaluation shows that our Relation Extraction module is able to capture a highly precise and compact set of weighted triples, and demonstrates the positive impact of the novel scoring metric we introduced. Moreover, we have shown that a high percentage of the knowledge encoded in our MKB is not present in other general and domain-specific KBs. Finally, regarding extrinsic evaluation, the Recommendation experiment confirms that explanations based on the learned KB are positively regarded by the users.

We identified several avenues for future work. For instance, we would like to extend our experiments to other Music datasets of varied registers (e.g. social networks, magazines, encyclopedias), in order to fully understand the core differences between this domain and standard language. This should give an approximate idea of whether we need specific tools in certain NLP tasks. For instance, it seems reasonable to think about a Music Entity Linking tool that is able to cope better with certain particularities of this domain. In addition, the identification of applications and the development of new methodologies in Music Information Retrieval that exploits MKBs is still an open problem. Finally, our ultimate end will be the creation of a broad and unified MKB that encodes most of the available music and musicological knowledge.

References

[1] F. M. Suchanek, G. Kasneci, G. Weikum, Yago: A core of semantic knowledge, in: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 697–706.

[2] C. Baral, G. De Giacomo, Knowledge representation and reasoning: Whats hot, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[3] E. Cambria, B. White, Jumping nlp curves: a review of natural language processing research [review article], Computational Intelligence Magazine, IEEE 9 (2) (2014) 48–57.

[4] G. A. Miller, Wordnet: A lexical database for english, Communications of the ACM 38 (11) (1995) 39–41.

[5] K. Degtyarenko, P. De Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcántara, M. Darsow, M. Guedj, M. Ashburner, Chebi: A database and ontology for chemical entities of biological interest, Nucleic acids research 36 (suppl 1) (2008) D344–D350.

[6] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, et al., Gene ontology: Tool for the unification of biology, Nature genetics 25 (1) (2000) 25–29.

[7] K. A. Spackman, K. E. Campbell, R. A. Côté, Snomed rt: A reference terminology for health care., in: Proceedings of the AMIA annual fall symposium, American Medical Informatics Association, 1997, p. 640.

[8] A. Swartz, Musicbrainz: A semantic web service, Intelligent Systems, IEEE 17 (1) (2002) 76–77.

[9] A. Freitas, J. C. da Silva, E. Curry, P. Buitelaar, Approximate and selective reasoning on knowledge graphs: A distributional semantics approach, Data & Knowledge Engineering 100 (2015) 211–225.

[10] K. Knight, S. K. Luk, Building a large-scale knowledge base for machine translation, in: AAAI, Vol. 94, 1994, pp. 773–778.

[11] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, N. A. Smith, Retrofitting word vectors to semantic lexicons, in: NAACL, 2014.

[12] I. Iacobacci, M. T. Pilehvar, R. Navigli, Sensembed: Learning sense embeddings for word and relational similarity, in: Proceedings of ACL, 2015, pp. 95–105.

[13] J. Camacho-Collados, M. T. Pilehvar, R. Navigli, Nasari: a novel approach to a semantically-aware representation of items, in: Proceedings of NAACL, 2015, pp. 567–577.

[14] E. H. Huang, R. Socher, C. D. Manning, A. Y. Ng, Improving word representations via global context and multiple word prototypes, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, Association for Computational Linguistics, 2012, pp. 873–882.

[15] A. Fader, L. Zettlemoyer, O. Etzioni, Open question answering over curated and extracted knowledge bases, Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14 (2014) 1156–1165.

[16] N. Aggarwal, P. Mika, R. Blanco, P. Buitelaar, Insights into Entity Recommendation in Web Search, Proceedings of the 4th International Workshop on Intelligent Exploration of Semantic Data.

[17] V. Ostuni, T. Di Noia, R. Mirizzi, E. Di Sciascio, A linked data recommender system using a neighborhood-based graph kernel, in: E-Commerce and Web Technologies, Vol. 188 of Lecture Notes in Business Information Processing, Springer International Publishing, 2014, pp. 89–100.

[18] N. Voskarides, E. Meij, Learning to Explain Entity Relationships in Knowledge Graphs, Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing 1 (2015) 564–574.

[19] L. Fang, A. A. D. Sarma, C. Yu, P. Bohannon, REX: Explaining Re-

lationships Between Entity Pairs, Proceedings of the VLDB Endowment (PVLDB) 5 (3) (2011) 241–252. arXiv:1111.7170.

[20] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, et al., Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia, Semantic Web Journal 5 (2014) 1–29.

[21] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, T. M. Mitchell, Toward an architecture for never-ending language learning., in: AAAI, Vol. 5, 2010, p. 3.

[22] N. Nakashole, G. Weikum, F. Suchanek, Patty: a taxonomy of relational patterns with semantic types, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Association for Computational Linguistics, 2012, pp. 1135–1145.

[23] R. Navigli, S. P. Ponzetto, Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artificial Intelligence 193 (2012) 217–250.

[24] C. D. Bovi, L. Espinosa-Anke, R. Navigli, Knowledge base unification via sense embeddings and disambiguation, in: Proceedings of EMNLP, 2015, pp. 726–736.

[25] C. Havasi, R. Speer, J. Alonso, Conceptnet 3: A flexible, multilingual semantic network for common sense knowledge, in: Recent Advances in Natural Language Processing, Citeseer, 2007, pp. 27–29.

[26] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, 2008, pp. 1247–1250.

[27] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni, Open information extraction for the web, in: IJCAI, Vol. 7, 2007, pp. 2670–2676.

[28] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni, Open Information Extraction from the Web, in: International Joint Conferences on Artificial Intelligence, 2007, pp. 2670–2676.

[29] A. Fader, S. Soderland, O. Etzioni, Identifying Relations for Open Information Extraction, in: Empirical Methods in Natural Language Processing, 2011.

[30] A. Carlson, J. Betteridge, R. C. Wang, E. Hruschka Jr, T. M. Mitchell, Coupled Semi-Supervised Learning for Information Extraction, in: Proc. of the third ACM WSDM, 2010, pp. 101–110.

[31] A. Moro, R. Navigli, Wisenet: Building a wikipedia-based semantic network with ontologized relations, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, ACM, New York, NY, USA, 2012, pp. 1672–1676.

[32] A. Moro, R. Navigli, Integrating syntactic and semantic analysis into the open information extraction paradigm, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 2148–2154.

[33] C. D. Bovi, L. Telesca, R. Navigli, Large-scale information extraction from textual definitions through deep syntactic and semantic analysis, Transactions of the Association for Computational Linguistics 3 (2015) 529–543.

[34] L. Tesnière, Elements de syntaxe structurale, Editions Klincksieck, 1959.

[35] L. Espinosa-Anke, H. Saggion, Applying dependency relations to definition extraction, in: Natural Language Processing and Information Systems, Springer, 2014, pp. 63–74.

[36] R. C. Bunescu, R. J. Mooney, A shortest path dependency kernel for relation extraction, in: Proc. of the conference on HLT/EMNLP, 2005, pp. 724–731.

[37] A. Culotta, J. Sorensen, Dependency tree kernels for relation extraction, in: Proc. of the 42Nd Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2004.

[38] P. Gamallo, M. Garcia, S. Fernández-Lanza, Dependency-based open information extraction, in: Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP, ROBUS-UNSUP '12, Association for Computational Linguistics, Stroudsburg, PA, USA, 2012, pp. 10–18.

[39] M. Sordo, S. Oramas, L. Espinosa-Anke, Extracting relations from unstructured text sources for music recommendation, in: Proceedings of NLDB 2015.

[40] M. S. Sergio Oramas, X. Serra, Automatic creation of knowledge graphs from digital musical document libraries, in: Conference in Interdisciplinary Musicology, Berlin, 2014.

[41] S. Oramas, F. Gómez, E. Gómez, J. Mora, Flabase: Towards the creation of a flamenco music knowledge base, in: 16th International Society for Music Information Retrieval Conference, 2015.

[42] O. Celma, X. Serra, FOAFing the music: Bridging the semantic gap in music recommendation, Web Semantics 6 (2008) 250–256. doi:10.1016/j.websem.2008.09.004.

[43] S. Oramas, M. Sordo, L. Espinosa-Anke, X. Serra, A semantic-based approach for artist similarity, in: 16th International Society for Music Information Retrieval Conference, Málaga, Spain, 2015.

[44] J. P. Leal, V. Rodrigues, R. Queirós, Computing Semantic Relatedness using DBPedia, 1st Symposium on Languages, Applications and Technologies, SLATE 2012.

[45] V. C. Ostuni, S. Oramas, T. D. Noia, X. Serra, E. D. Sciascio, A Semantic Hybrid Approach for Sound Recommendation, 24th International World Wide Web Conference (WWW 2015) (2015) 3–4.

[46] Ò. Celma, P. Herrera, A new approach to evaluating novel recommendations, in: Proceedings of the 2008 ACM conference on Recommender systems, ACM, 2008, pp. 179–186.

[47] A. Passant, dbrec - Music Recommendations Using DBpedia, in: The Semantic Web–ISWC, Vol. 1380, Springer, 2010, pp. 209–224.

[48] B. Bohnet, Very high accuracy and fast dependency parsing is not a contradiction, in: Proceedings of the 23rd International Conference on Computational Linguistics, 2010, pp. 89–97.

[49] P. N. Mendes, M. Jakob, A. García-silva, C. Bizer, DBpedia Spotlight : Shedding Light on the Web of Documents, Proc. of the 7th International Conference on Semantic Systems.

[50] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: a unified approach, Transactions of the Association for Computational Linguistics 2 (2014) 231–244.

[51] P. Ferragina, U. Scaiella, Tagme: on-the-fly annotation of short text fragments (by wikipedia entities), in: Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, 2010, pp. 1625–1628.

[52] Mausam, M. Schmitz, R. Bart, S. Soderland, O. Etzioni, Open Language Learning for Information Extraction, in: Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012.